RasPi

DESIGN
BUILD
CODE

17

Get hands-on with your Raspberry Pi

Make your own

SAT-NAV

PLUS

HOW TO
PAINT
CIRCUITS

Welcome

With the holidays just around the corner, it's high time for a big, somewhat ambitious project – building your own sat-nav! Actually, there's a little more to it than that, since we're also going to be adding in a music player and pulling in weather reports using a GPS module. And the Navit-powered maps will be voice-controlled too, of course. The great thing with this project is that you can extend it however you like – for example, adding in video-playing functions to turn it into more of a car infotainment system. We've also got some great guides to securing your Raspberry Pi and powering-up older models with some clever optimisation tricks. Oh, and we're making J.A.R.V.I.S. as well… Have fun!

Gavin Thomas

Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

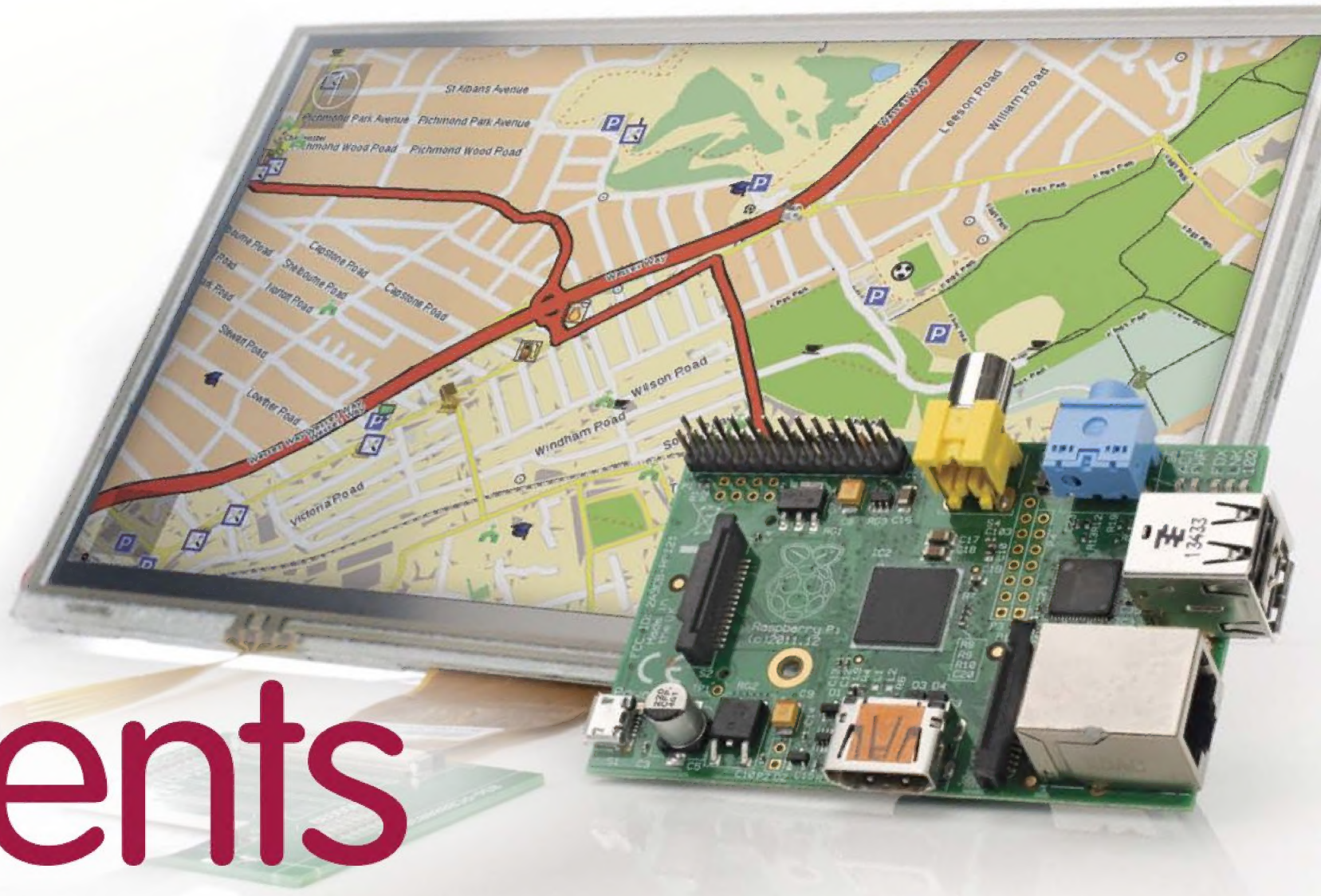
Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

Join the conversation at…

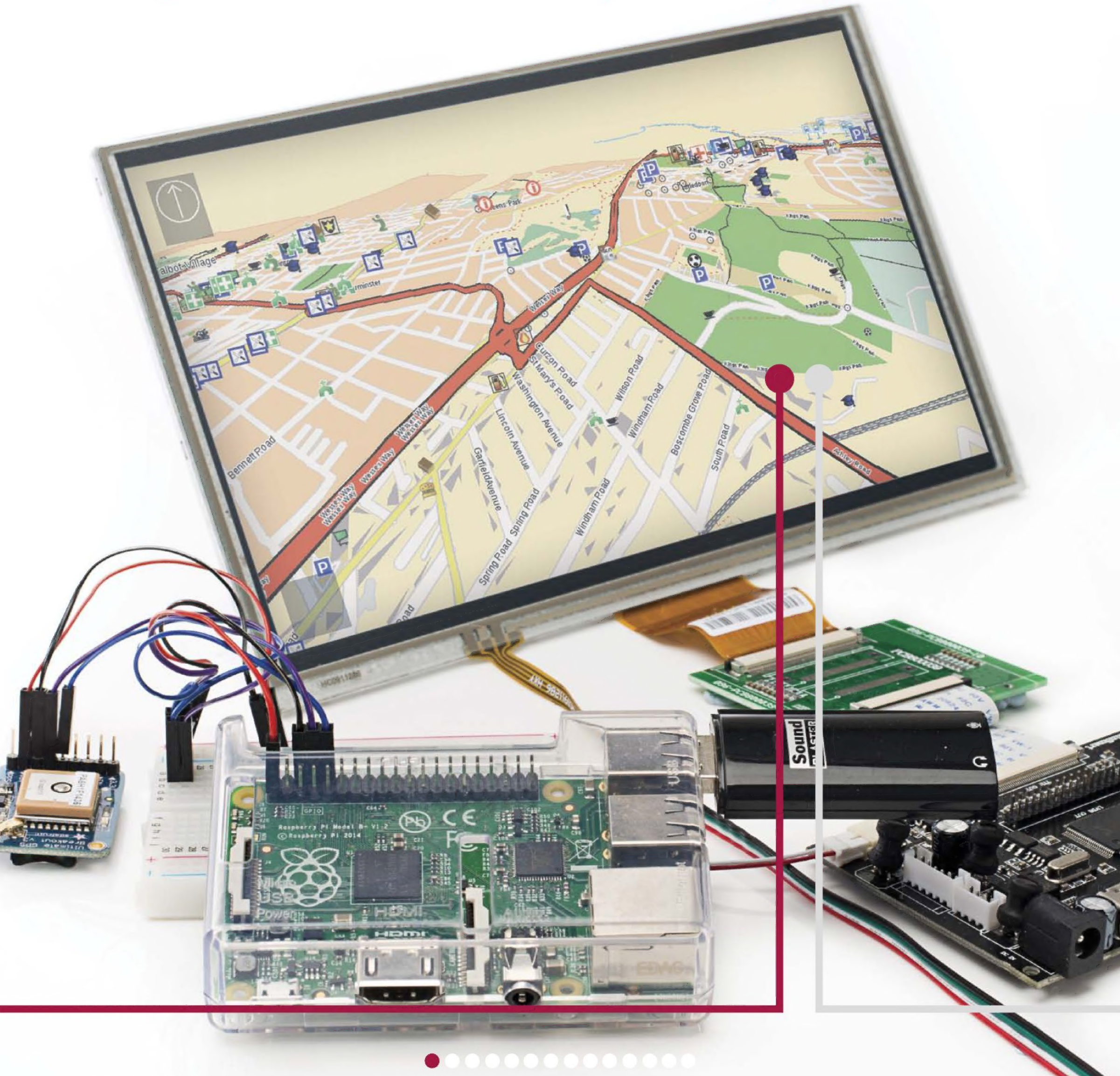
@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk



Contents

Raspberry Pi Sat-Nav

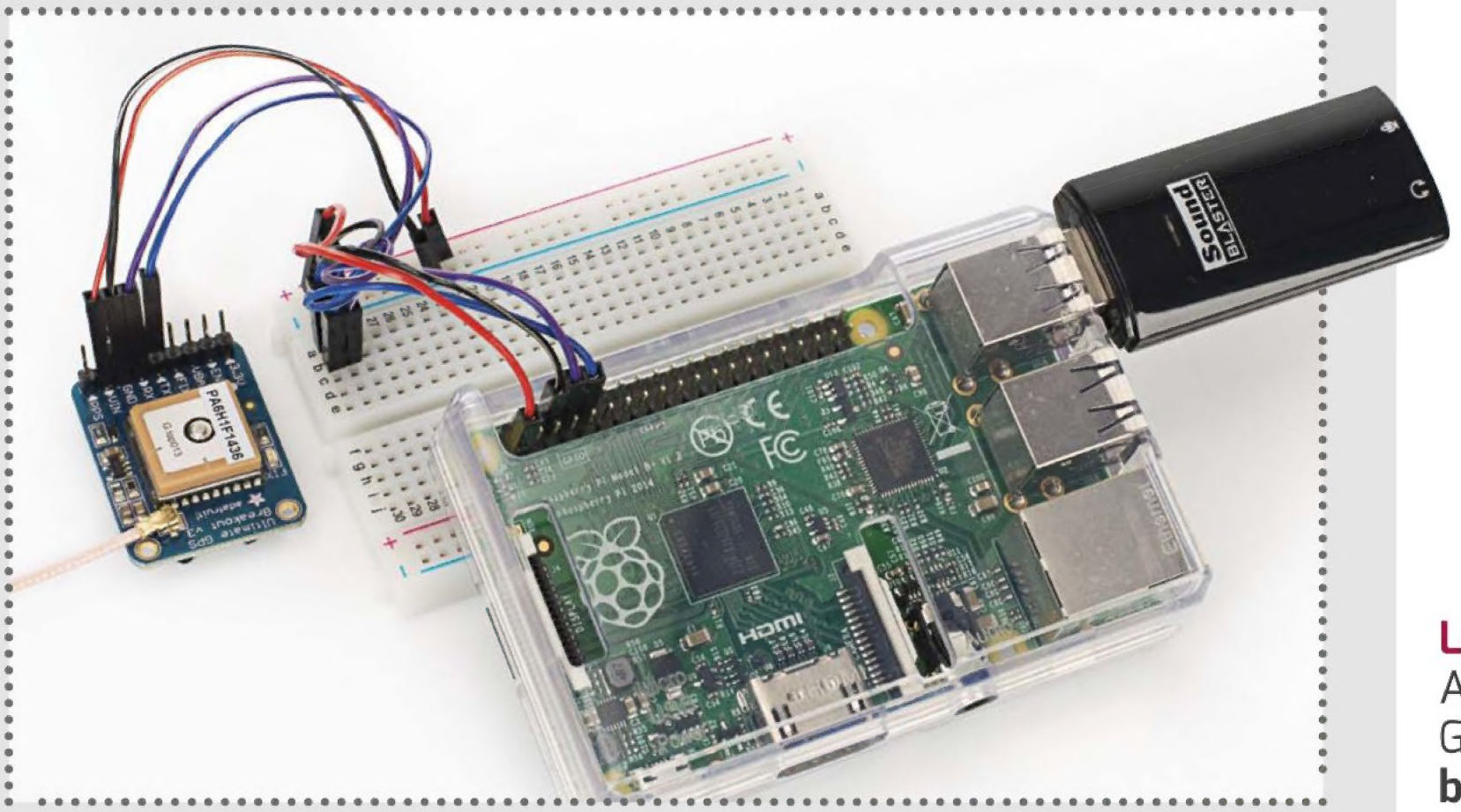
Make your own touchscreen navigation system that gives directions, weather reports and plays music



Cars are getting clever. These days, with smart navigation interfaces built into new cars, you don't need to buy a TomTom to get help with directions. But if you've got a Raspberry Pi then you don't even need to buy that – let alone a new car!

In this project we will show you how to build your own car computer using your Pi, a touchscreen like the 9-inch model from SainSmart that we're using here, and a few other bits like a GPS module and USB 3G modem. Your CarPi will be able to use open source navigation software Navit to show your route map on-screen, plus speech synthesis to read out directions, and it will also be able to check your location and give you weather reports. It'll work as a music player too, of course.

It's an ambitious project, but you will gain a solid understanding of custom-made interfaces, navigation software and geolocation data, touchscreen calibration, speech synthesis and more. And you don't have to use the same SainSmart screen as us – you can use your official Raspberry Pi 7-inch Touchscreen Display. Check out the components list to the right, make sure you've got everything and then let's get started!



Left We're using Adafruit's excellent GPS Breakout kit here: **bit.ly/1G8X2gw**

THE PROJECT ESSENTIALS

Soldering iron

Female-to-female jumper cables

Adafruit GPS Ultimate Breakout Board

Adafruit GPS SMA-to-uFL adapter cable

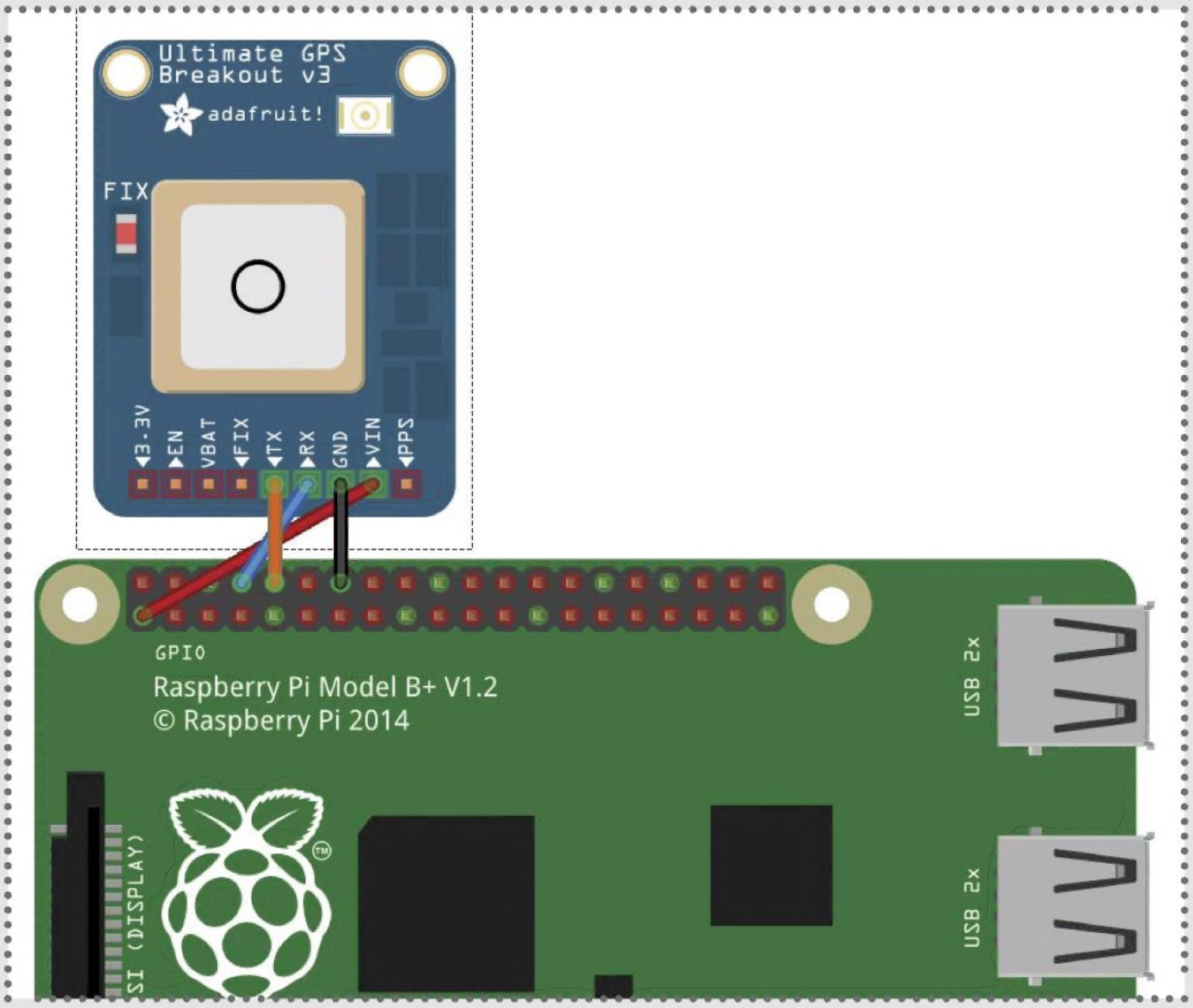
Adafruit GPS SMA antenna (3-5V, 28dB, 5m)

External USB sound card (optional)

USB 3G modem (optional)

Touch Screen (e.g. SainSmart HDMI/VGA 9-inch Touch Screen LCD + Driver Board)

12V Power supply for screen (a car charger for laptops with a USB port is ideal)



Left Adafruit has a great library for its Ultimate GPS Breakout –check out **http://bit.ly/1l7tASN**

01 Basic configuration

Boot up your Raspberry Pi and expand the filesystem using raspi-config. Go to Advanced Options and disable the Serial connection – you'll need this to talk to the GPS module later. In raspi-config, enable X at boot as the pi user. Say Yes to reboot. Once rebooted, ensure your packages are up to date with:

```
sudo apt-get update
sudo apt-get upgrade
```

02 Connect GPS module

Solder the pin headers onto the Adafruit GPS module. You can also solder the battery connector which is used to keep the device partially active, giving a faster fix. You only need to use 4 pins: 3.3V, ground, serial transmit and serial receive. Power the Pi off again before connecting anything.

"You can also solder the battery connector which is used to keep the device partially active, giving a faster fix"

As we are using GPS, the antenna will have to go outside or under a window in order to gain signal. Connect the antenna to the board and power everything back on. The light on the GPS module will flash frequently while finding a fix. Once it has one, it will blink every 15 seconds.

03 Install navigation software

Begin to install the Navit navigation software by entering the following commands:

```
sudo apt-get install navit gpsd gpsd-clients
espeak
sudo nano /etc/default/gpsd
set START_DAEMON="true"
```

...and then set:

```
DEVICES="/dev/ttyAMA0"
```

Start the GPS daemon with:

```
sudo /etc/init.d/gpsd start
```

You can check that it's working by looking at the GPS data with:

```
cgps -s
```

04 Connect the screen

The SainSmart screen doesn't come with any written instructions. Instead, there is a YouTube video on SainSmart's website with details about how to put it together: **http://bit.ly/1DF6eJJ**. The important part is that the DC power supply should be 12V.

05 Set the screen resolution

We will have to force the correct resolution (1024x600) for the screen by editing /boot/config.txt with `sudo nano`. To do so, add the following options:

"The light on the GPS module will flash frequently while finding a fix. Once it has one, it will blink every 15 seconds"

```
framebuffer_width=1024
framebuffer_height=600
hdmi_force_hotplug=1
hdmi_cvt=1024 600 60 3 0 0 0
hdmi_group=2
hdmi_mode=87
```

For the changes to properly take effect, you will need to reboot with `sudo reboot`.

06 Download kernel source

To start the touchscreen, you need to compile an extra kernel module to support it. The program rpi-source (**http://github.com/notro/rpi-source/wiki**) will find the source of your kernel. Install rpi-source with:

```
sudo wget https://raw.githubusercontent.com/
notro/rpi-source/master/rpi-source -O usr/
bin/rpi-source && sudo chmod +x /usr/bin/rpi-
source && /usr/bin/rpi-source -q -tag-update
```

…then run rpi-source in order to get the source of the running kernel.

07 Update GCC

As we write this, recent Raspberry Pi kernels are compiled with GCC 4.8 and Raspbian only comes with 4.6, so you will have to install 4.8 to continue with the following steps. Do this by entering:

```
sudo apt-get install -y gcc-4.8 g++-4.8
ncurses-dev
```

Then you have to set GCC 4.8 as the default:

```
sudo update-alternatives --install /usr/bin/
gcc gcc /usr/bin/gcc-4.6 20
sudo update-alternatives --install /usr/bin/
gcc gcc /usr/bin/gcc-4.8 50
```

Embed the screen

We've looked at the PiTFT and the HDMIPi before, but the SainSmart touchscreen we're using here is well suited to many embedded projects. It's larger than the PiTFT but also without the large bezels of the HDMIPi – and it's incredibly thin, so useful for installations like a live photo frame or a home automation control interface embedded into a cupboard door.

```
sudo update-alternatives --install /usr/bin/
    g++ g++ /usr/bin/g++-4.6 20
sudo update-alternatives --install /usr/bin/
    g++ g++ /usr/bin/g++-4.8 50
```

08 Pick the module to compile

Rpi-source puts the kernel source in a folder called 'linux'. To choose the USB Touchscreen Driver, enter:

```
cd linux
make menuconfig
Device Drivers ->  Input device support ->
    Generic input layer (needed for keyboard,
    mouse, …)  -> Touchscreens (press space to
    include)  -> USB Touchscreen Driver (press
    M to make module)
```

Once you've done that, you then need to make sure you save your changes as '.config' and run scripts/diffconfig to see the differences.

09 Compile and install the module

Now you need to compile and install the module. Do so by entering:

```
make prepare
make SUBDIRS=drivers/input/touchscreen modules
sudo make SUBDIRS=drivers/input/touchscreen
    modules_install
sudo depmod
```

If you unplug and reconnect the touchscreen, it should work fine but it will probably need calibrating.

10 Calibrate the touchscreen

At this point, you can easily calibrate the touchscreen by entering the following:

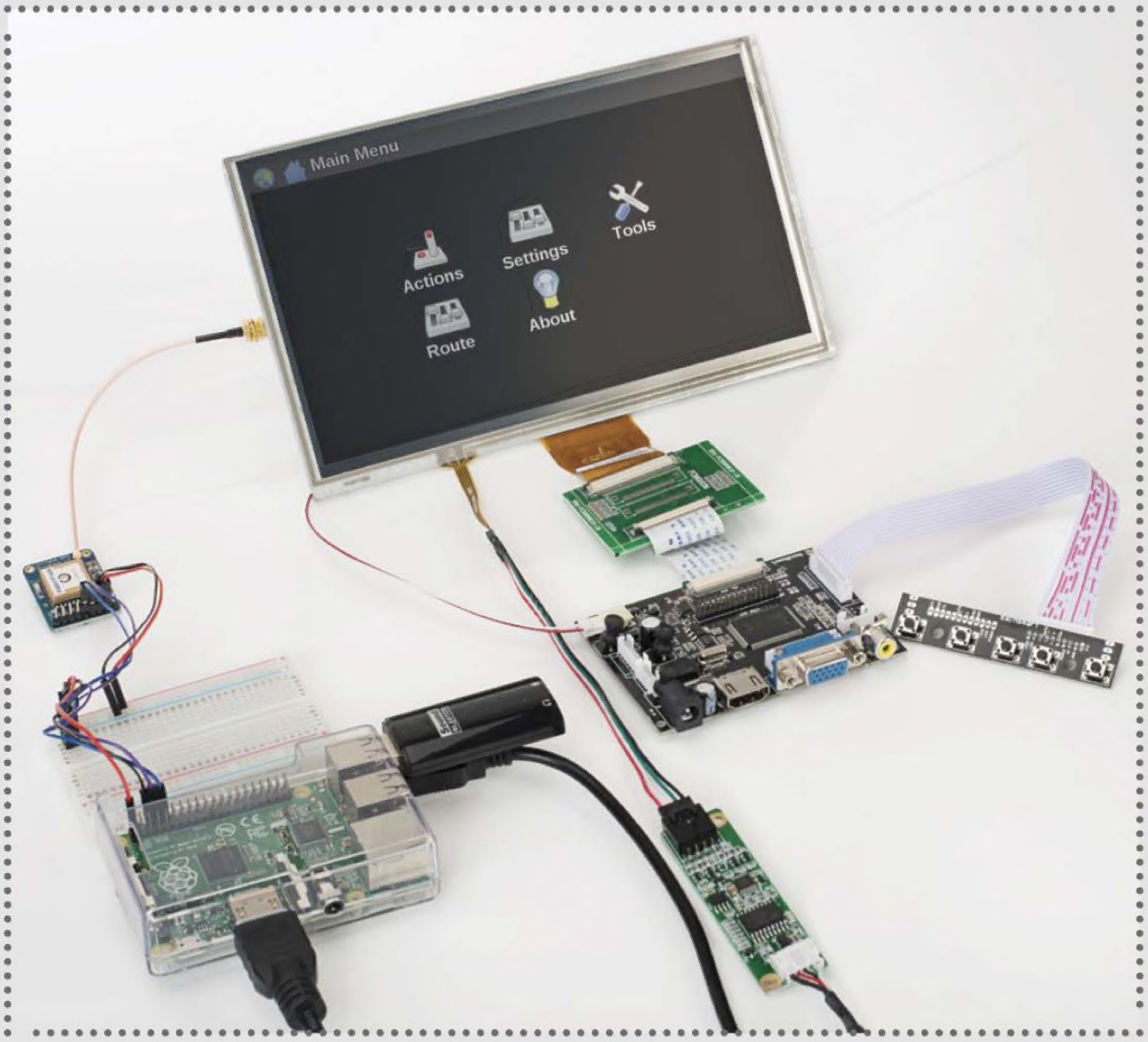
"Make sure you save your changes as '.config' and run scripts/diffconfig to see the differences"

```
cd /etc/X11
sudo mkdir xorg.conf.d
cd xorg.conf.d
sudo nano 99-calibration.conf
```

...with the following content:

```
Section "InputClass"
    Identifier "calibration"
    MatchProduct "eGalax Inc. USB TouchController"
    Option    "SwapAxes"    "1"
    Option    "InvertX"     "1"
EndSection
```

"InvertX" actually inverts Y because the axes have been swapped around. Reboot your Pi again for these changes to occur. Now the calibration is roughly correct, download an input calibrator that Adafruit has packaged already:



Left The setup is a little complex, so lay it all out on your workbench first to ensure it's wired and soldered correctly

```
wget http://adafruit-download.s3.amazonaws.com/
    xinput-calibrator_0.7.5-1_armhf.deb
sudo dpkg -i xinput-calibrator_0.7.5-1_armhf.
    deb
DISPLAY=:0.0 xinput_calibrator
```

DISPLAY=:0.0 is useful because you can run the program from any terminal (including an SSH session) and have it appear on the touchscreen. Touch the points that appear on the screen as prompted. Once the program is finished, you should get an output that is similar to the following:

```
Option  "Calibration"    "84 1957 270 1830"
```

Add this to the '99-calibration.conf' file that we created earlier, just below the other Option entries.

11 Download maps

Navit needs maps, so go and download these from **http://maps.navit-project.org**. You can either use the web browser on the Pi or download the map from another machine and copy it using `scp`. Use the predefined area option to select where you live. The smaller the area that you pick, the less data you will have to process. Here, the UK has a map size of 608 MB. Now move the map to the navit folder:

```
mkdir -p /home/pi/.navit/maps
mv /home/pi/Downloads/$your_map/home/pi/.
    navit/$country.bin
```

For example:

```
mv /home/pi/Downloads/osm_bbox_9.7,49.6,2.2,
    61.2.bin/home/pi/.navit/maps/UK.bin
```

12 Navit configuration

Sudo-edit /etc/navit/navit.xml with your favourite editor.

> "DISPLAY=:0.0 is useful because you can run the program from any terminal (including an SSH session) and have it appear on the touchscreen"

Search for "openstreetmaps". Now disable the sample map above, enable the openstreetmap mapset and set the data variable to where you just moved your map. In this case, it looks like this:

```
<!-- Mapset template for openstreetmaps -->
<mapset enabled="yes">
<map type="binfile" enabled="yes" data="/home/
    pi/.navit/maps/UK.bin"/>
</mapset>
```

Then search for osd entries similar to:

```
<osd enabled="yes" type="compass"/>
```

...and enable the ones you want – we recommend enabling them all. You may want to zoom in closer than the default map layout. A zoom value of 64 is useful.

13 Sound configuration

Before configuring speech support for Navit, configure the external sound card. You have to stop the Broadcom module from loading and remove some Raspberry Pi-specific ALSA (Advanced Linux Sound Architecture). To do this, sudo-edit /etc/modprobe and comment out (i.e. prefix with a #) this line:

```
snd-bcm2835
```

Then run:

```
sudo rm /etc/modprobe.d/alsa*
```

Reboot for the changes to take effect. Use alsamixer to set the volume on the if it's too quiet.

14 Download a voice

The speech synthesis software needs a voice and a proprietary binary. You can get both by completing the following steps:

Make it mobile

While it's best to put this all together in a clean space initially, the point of the project is to put it in your car. You could install everything into a hand-made enclosure or modify a bought one, or you could secure the various parts inside, for example, your glovebox or car doors. You'll also need to power the screen and Pi with a power pack and ensure the GPS antenna is getting a decent signal.

```
sudo mkdir -p /usr/share/mbrola/voices/
wget http://www.tcts.fpms.ac.be/synthesis/
    mbrola/dba/en1/en1-980910.zip
unzip en1-980910.zip
sudo cp en1/en1 /usr/share/mbrola/voices
wget http://www.tcts.fpms.ac.be/synthesis/
    mbrola/bin/raspberri_pi/mbrola.tgz
tar zxvf mbrola.tgz
sudo mv mbrola /usr/local/bin/
```

15 Create speech script

Navit supports speech by running an external script and passing the text to speak as an argument. Create one using:

```
cd /home/pi/.navit
wget http://liamfraser.co.uk/lud/carpi/chime.wav
touch speech.sh
chmod +x speech.sh
```

Now edit speech.sh:

```
#!/bin/bash
aplay -r 44100 /home/pi/.navit/chime.wav
espeak -vmb-en1 -s 110 -a 150 -p 50 "$1"
```

Finally, test it with:

```
./speech.sh "Hello World"
```

16 Configure Navit for speech

The last part is simple. Edit the Navit config file again (/etc/navit/navit.xml) and replace the following line:

```
<speech type="cmdline" data="echo 'Fix the
    speech tag in navit.xml to let navit say:'
    '%s'" cps="15"/>
```

...with:

```
<speech type="cmdline" data="/home/pi/.navit/
    speech.sh %s" cps="10" />
```

"Navit supports speech by running an external script and passing the text to speak as an argument"

Left The Navit software comes with a host of options built into its menu system

Now you can run Navit with `DISPLAY=:0.0 navit` and have fun experimenting.

17 Install the music player

MPD is the music player back-end and pympdtouchgui is the front-end that needs installing manually:

```
sudo apt-get install mpd ncmpcpp
wget http://www.spida.net/projects/software/
    pympdtouchgui/pympdtouchgui-0.320.tgz
tar zxvf pympdtouchgui-0.320.tgz
cd pympdtouchgui-0.320/
sudo python setup.py install
# Fix hardcoded path in software
sudo ln -s /usr/local/share/pympdtouchgui/ /
    usr/share/pympdtouchgui
```

18 Copy music

Scp (secure copy protocol) was used here to copy music. First get the Pi's IP address by running `ip addr`. Then run `sudo passwd` to set a password for root. From a computer with music on, run:

```
scp -r music_folder root@pi_ip_address:/var/
    lib/mpd/music/
```

Then on the Pi, change the ownership of the music that you just copied:

```
sudo chown -R mpd:audio /var/lib/mpd/music
```

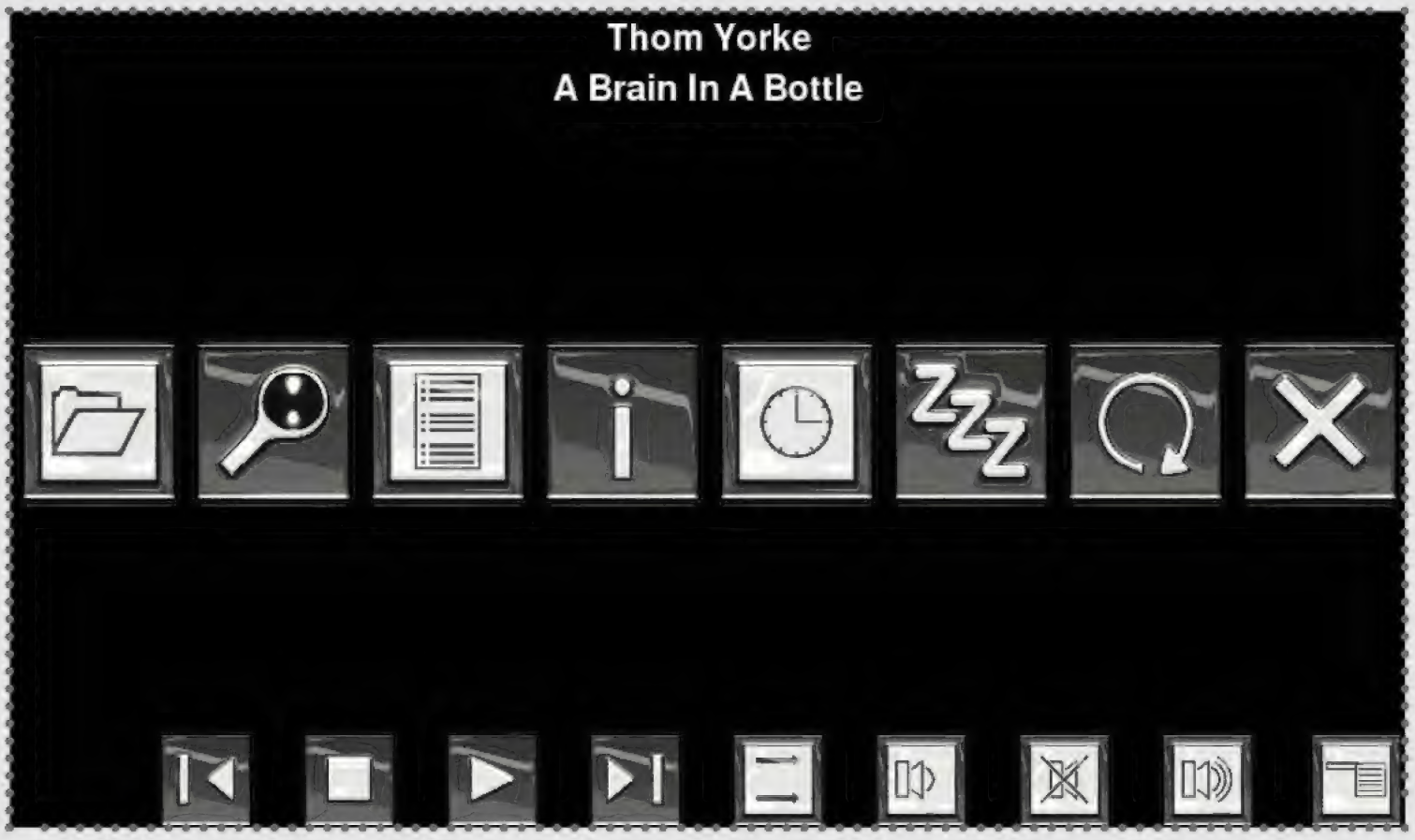
19 Update mpd music library

Ncmpcpp is a command line client for mpd. Type `ncmpcpp` and press U to update the library. Press 3 to browse the library and check the music is there, and press Q to quit. Pressing 1 will select the help screen if you want to do more.

20 Install awesome window manager

Now you will need to write your own launcher for CarPi, which will run full-screen. To ensure every application is forced to full-screen, use awesome window manager in full-screen mode.

```
sudo apt-get install awesome
sudo rm /etc/alternatives/x-session-manager
sudo ln -s /usr/bin/awesome /etc/alternatives/
    x-session-manager
```



Left The pympdtouchgui front-end for the music player is surprisingly full of features

When changing the default x-session-manager, awesome will be auto-started at boot instead of LXDE. If you reboot the Pi, awesome should then load up automatically.

21 Install the launcher requirements

The launcher is going to use a weather API combined with location data from the GPS receiver to give weather updates when requested. The nicest HTTP API for Python is requests, which you can install by doing the following:

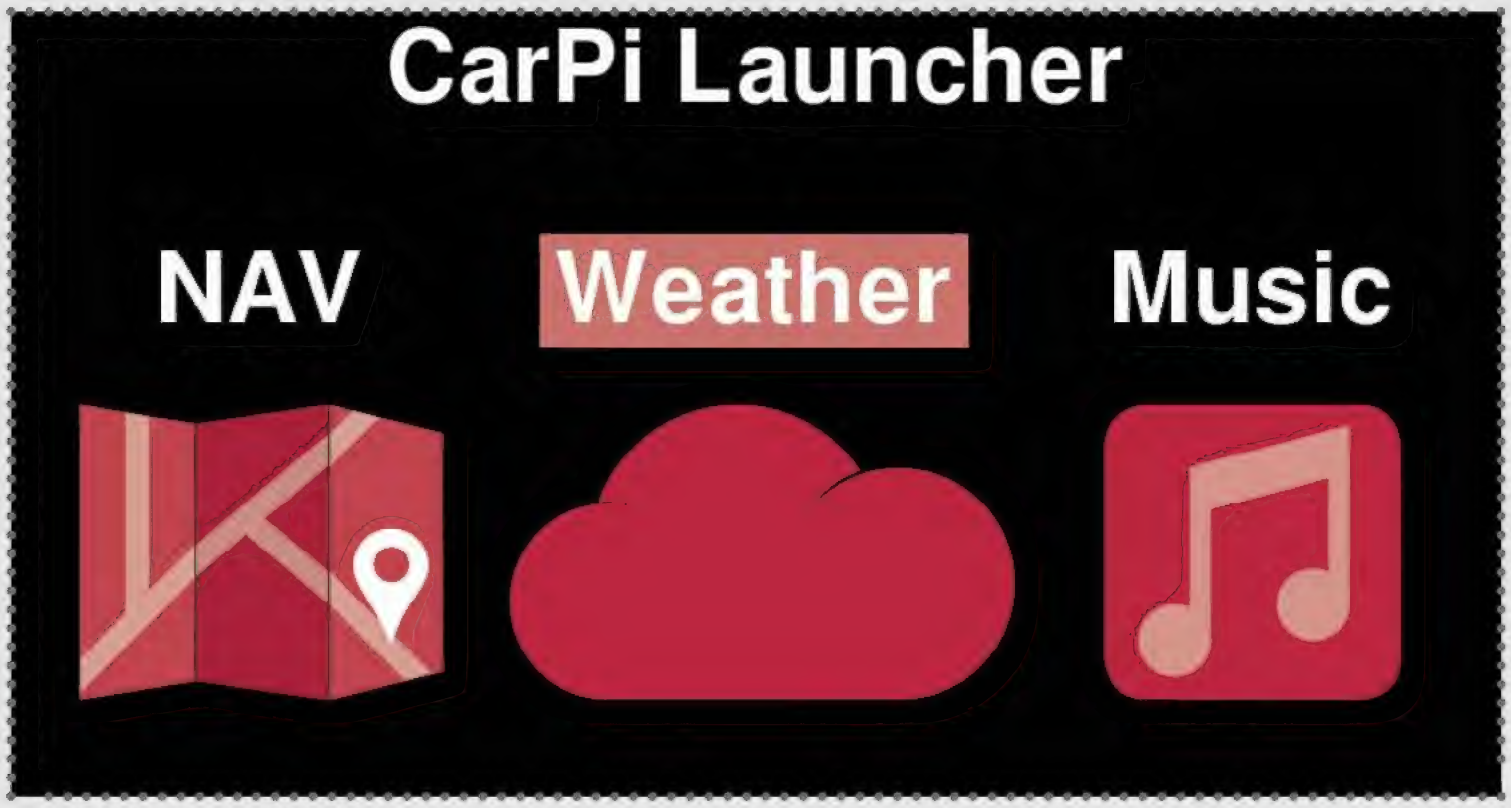
```
sudo apt-get install python-pip
sudo pip install requests
```

22 Write the launcher code

Creating the code itself is pretty self explanatory, but you can use our ready-made version by downloading the CarPi package from **http://bit.ly/1MEfOy6** and extracting carlauncher/carlauncher.py.

23 Start the launcher automatically

Sudo-edit /etc/xdg/awesome/rc.lua and move awful.layout.suit.max.fullscreen to the top of the layouts list.



Left Our launcher is a simple, text-only affair – try upgrading it with custom icons like these

"The launcher is going to use a weather API combined with location data from the GPS receiver to give weather updates when requested"

Add the following to the bottom of the file:

```
awful.util.spawn_with_shell("/home/pi/
carlauncher/carlauncher.py")
```

Now reboot again and the launcher should come up automatically this time.

24 Future improvements

There are a number of improvements that could be made to the base project at this point:

- Make the launcher switch between applications rather than start them again each time
- Make the launcher look better aesthetically by adding icons
- Use Mopidy instead of MPD so you can use Spotify
- Further Navit configuration to make it more featureful
- Add an SSD or USB flash drive for storage to make things quicker

Below Navit looks great on the touchscreen once it's all set up!

The Code SAT-NAV

```
#!/usr/bin/env python2
import os, sys, requests, pygame
from gps import *
from pygame.locals import *

class WeatherClient:
    apikey = "7232a1f6857090f33b9d1c7a74721"

    @staticmethod
    def latlon():
        gpsd = gps(mode=WATCH_ENABLE)

    # Needs better error handling
        try:
            while True:
                report = gpsd.next()
                if report['class'] == 'TPV':
                    gpsd.close()
                    return report['lat'], report['lon']
        except:
            return None, None

    @staticmethod
    def usefuldata(j):
        # Returns a string of useful weather data from a LOT of json
        d = j['data']['current_condition'][0]
        out = "Now - Temp: {0}C, Feels Like: {1}C, Description: {2}\n"\
            .format(d['temp_C'],
                    d['FeelsLikeC'],
                    d['weatherDesc'][0]['value'])

        hourly = j['data']['weather'][0]['hourly']
        hour_count = 1
        for h in hourly:
            out += ("+{0}hr - Temp: {1}C, Feels Like: {2}C, Chance of Rain:"
```

The Code SAT-NAV

```
                    " {3}%, Description: {4}\n")\
                    .format(hour_count,
                            h['tempC'],
                            h['FeelsLikeC'],
                            h['chanceofrain'],
                            h['weatherDesc'][0]['value'])
        hour_count += 1

        # Rstrip removes trailing newline
        return out.rstrip()

    @staticmethod
    def update():
        errstr = "Error getting weather data"

        lat, lon = WeatherClient.latlon()
        if lat == None or lon == None:
            return errstr
        api_req = ("http://api.worldweatheronline.com/free/v2/weather.ashx"
        "?q={0}%2C{1}&format=json&key={2}").format(lat, lon,
                                                  WeatherClient.apikey)
        r = None

        try:
            r = requests.get(api_req)
        except requests.exceptions.RequestException as e:
            return errstr
        return WeatherClient.usefuldata(r.json())

class CarLauncher:
    def __init__(self):
        pygame.init()
        pygame.mixer.quit() # Don't need sound
        screen_info = pygame.display.Info()
        self.screen = pygame.display.set_mode((screen_info.current_w,
```

The Code SAT-NAV

```
                                                    screen_info.current_h))
        pygame.display.set_caption('Car Launcher')
        self.titlefont = pygame.font.Font(None, 100)
        self.wfont = pygame.font.Font(None, 30)
        self.w_text = None # Weather text

    def clean_background(self):
        background = pygame.Surface(self.screen.get_size())
        self.background = background.convert()
        self.background.fill((0, 0, 0))

        # Render title centred
        text = self.titlefont.render("CarPi Launcher", 1, (255, 255, 255))
        textpos = text.get_rect()
        textpos.centerx = self.background.get_rect().centerx
        self.background.blit(text, textpos)
        self.screen.blit(self.background, (0,0))
        pygame.display.flip()

    def main_menu(self):
        # btns maps Text -> Rectangles we can do collision detection on
        self.btns = {'Music' : None, 'NAV' : None, 'Weather' : None}

        item_num = 1
        for key in self.btns:
            text = self.titlefont.render(key, 1, (255,255,255))
            textpos = text.get_rect()
            max_width = self.background.get_rect().width / len(self.btns)
            center_offset = max_width * 0.5
            # This y pos puts buttons just below title
            textpos.centery = self.background.get_rect().centery / 2
            textpos.centerx = (max_width * item_num) - center_offset
            self.btns[key] = textpos
            self.screen.blit(text, textpos)
            item_num += 1
```

The Code SAT-NAV

```
        pygame.display.flip()

    def select_rect(self, rect, text):
        # Colour a rect the user has clicked in green
        surface = pygame.Surface((rect.w, rect.h))
        surface.fill((0, 255, 0))
        # Now we have to draw the text over it again
        t = self.titlefont.render(text, 1, (255,255,255))
        surface.blit(t, (0,0))
        self.screen.blit(surface, rect)
        pygame.display.flip()

    def reset(self):
        self.clean_background()
        self.main_menu()
        self.render_weather()

    def execute(self, path):
        os.system(path)
        # os.system blocks, so by the time we get here the
        # application has finished
        self.reset()

    def render_weather(self):
        if self.w_text == None:
            return

        # Get y starting at the bottom of the nav button
        margin = 10
        y = self.btns['NAV'].bottomleft[1] + margin

        for t in self.w_text.split("\n"):
            line = self.wfont.render(t.rstrip(), 1, (255,255,255))
            line_rect = line.get_rect()
            line_rect.centerx = self.background.get_rect().centerx
```

The Code SAT-NAV

```
            line_rect.y = y
            self.screen.blit(line, line_rect)
            y += margin + line_rect.height
        pygame.display.flip()

    def handle_events(self, events):
        for e in events:
            if e.type == QUIT:
                sys.exit()
            elif e.type == MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos()
                # Check if it collides with any of the buttons
                for btn_text, rect in self.btns.iteritems():
                    if rect.collidepoint(pos):
                        self.select_rect(rect, btn_text)
                        if btn_text == "NAV":
                            self.execute("/usr/bin/navit")
                        elif btn_text == "Music":
                            self.execute("/usr/local/bin/pympdtouchgui")
                        elif btn_text == "Weather":
                            self.w_text = WeatherClient.update()
                            # Reset will render weather if string is populated
                            self.reset()

    def loop(self):
        clock = pygame.time.Clock()
        self.reset()
        while 1:
            self.handle_events(pygame.event.get())
            # 5 fps is plenty
            clock.tick(5)

if __name__ == "__main__":
    cl = CarLauncher()
    cl.loop()
```

Draw circuits with Bare Conductive paint

Assembling circuits with conductive paint enables you to combine art and electronics in a whole new way

Playing with electronics and physical computing is a very rewarding task. For a beginner, though, the mess of wires and components can become very confusing quite quickly, and soldering can be a safety concern when children are involved. Bare Conductive has taken the joy of electronics and made it far safer, easier and more versatile with its conductive paint. You can draw wires with a paintbrush, use it for cold-soldering or a conductive adhesive, and much, much more. There aren't many boundaries to what you can achieve – even multi-layer circuits are possible. Pair this paint with a microcontroller board and you could be creating interactive art, clothing and projects in no time.

THE PROJECT ESSENTIALS

Bare Conductive paint (pen or tub)

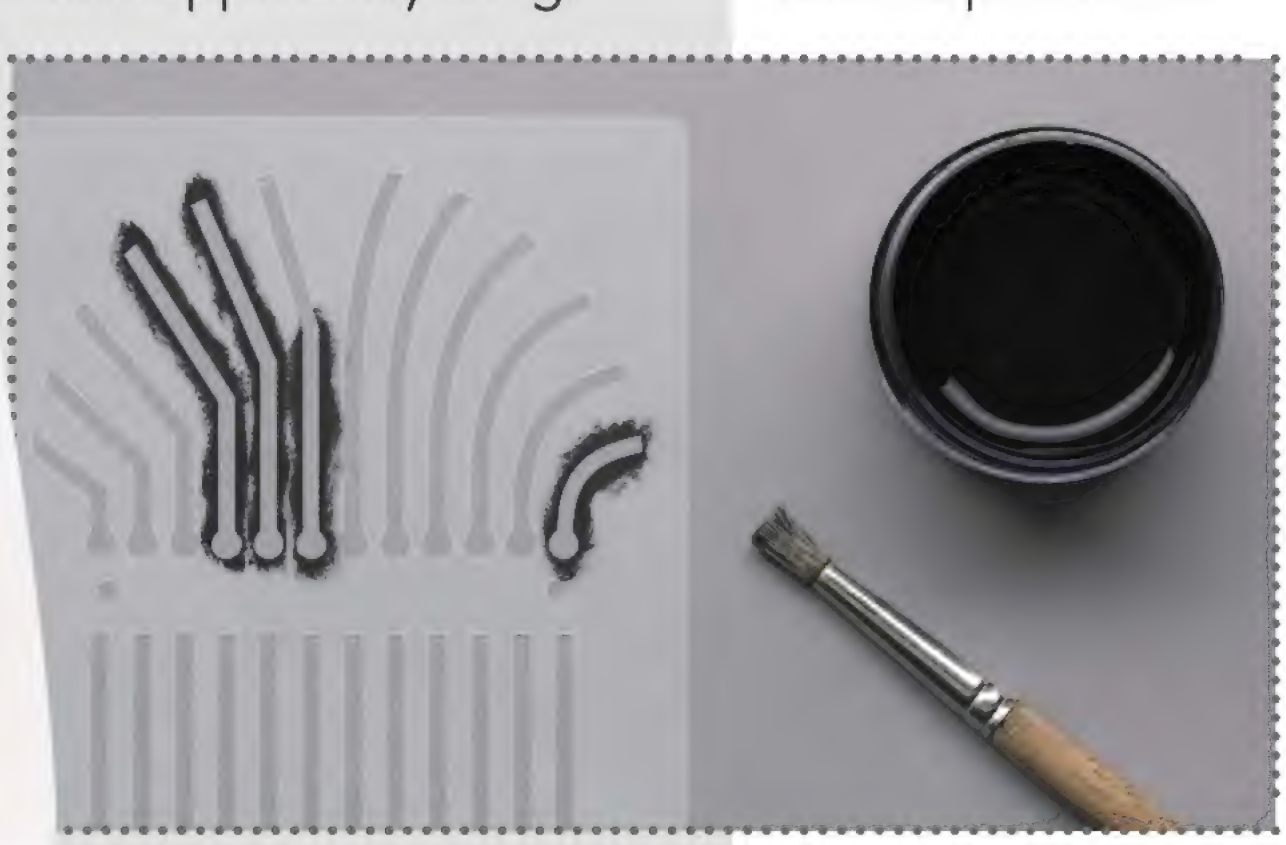
Male-to-female jumper cables

LEDs and resistors (optional)

01 Get your tools

Paint and a paintbrush aren't the first items that come to mind when you think about electronics, so you may be wondering where to get them from. Bare Conductive stocks the paint and a selection of components in its shop (**www.bareconductive.com/shop**) but you will need to go somewhere else for art supplies. Try a high street craft shop such as Hobbycraft (or go online: **www.hobbycraft.co.uk**).

Below Cut out custom templates to suit your project's style and build requirements



TRIC PAINT

02 Pick your platform

The great thing about Bare Conductive paint is that, when dry, it works just like normal wiring! That means you can use it with any of your favourite microcontrollers like the Bare Conductive Touch Board, a Raspberry Pi or Adafruit's wearable FLORA platform. Or you can just use some small pin batteries and flashing LEDs for a standalone system.

03 Start to paint

You can paint Bare Conductive paint onto pretty much any surface – paper, fabric, walls, clothing, wood, plastic and much more. For really accurate shapes and results, the best idea is to create or purchase a stencil (paper stencils are the easiest to make at home, but use vinyl for the best edge finish).

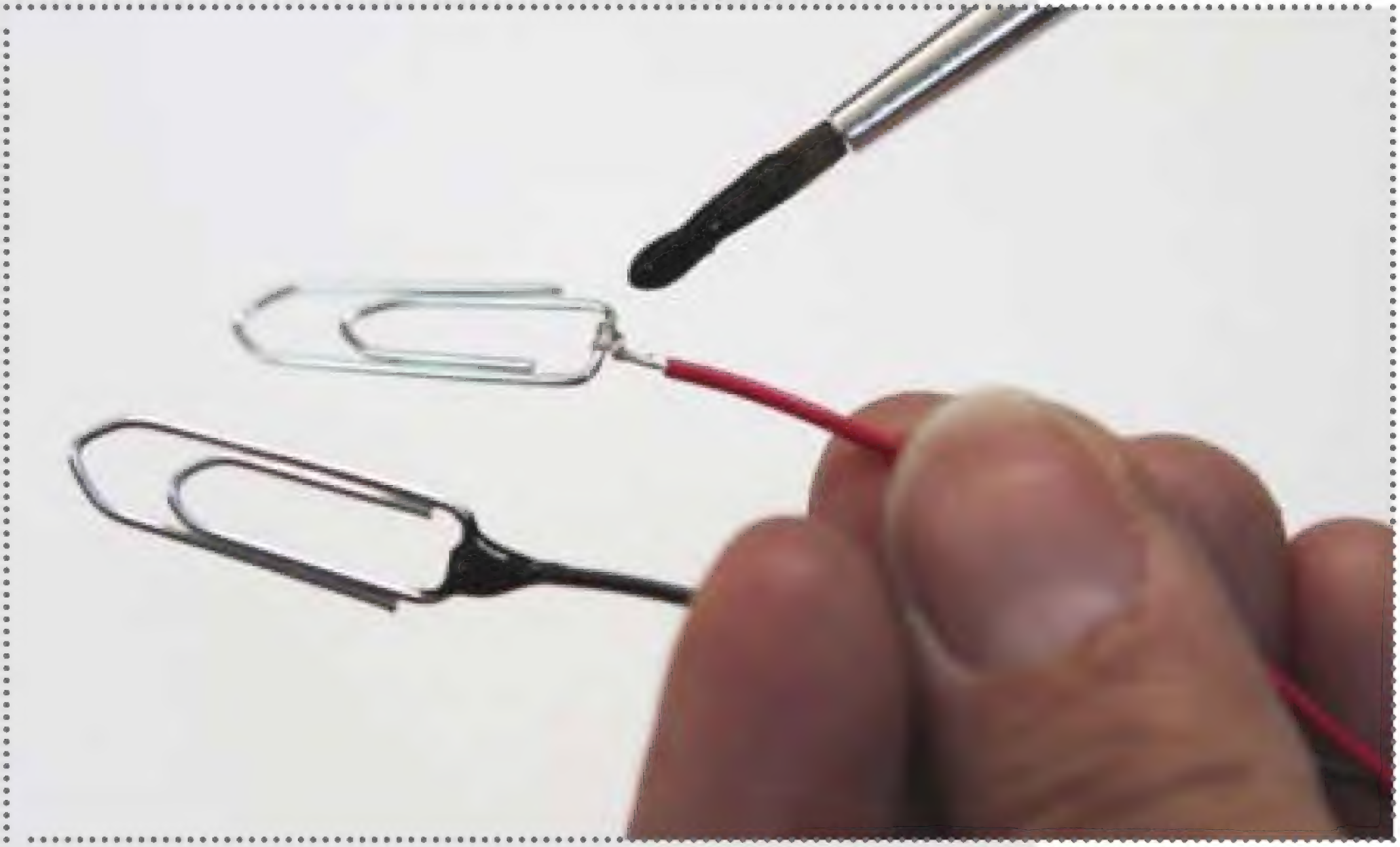
04 Connect it up

There are plenty of ways to connect to the conductive paint (from battery packs or microcontrollers, for example) no matter what surface it's on, because once it is dry it acts just like an uninsulated wire. This means you can use wires glued on with the paint, paper clips, bulldog clips, alligator clips or even sewn-in conductive snaps for wearables projects.

05 Make repairs

The conductive paint is thick and when it's dry it becomes quite strong. These means you can use it to cold-solder things together and repair any breakages. In other words, you could glue components into a circuit board or glue wires together and they would still function electrically. You can even use it to repair damaged tracks on circuit boards.

"You can paint Bare Conductive paint onto pretty much any surface – paper, fabric, walls, clothing, wood, plastic and much more"



06 Clean up

A lot of you are probably thinking that something as cool as conductive paint is going to be nasty stuff. Actually, Bare Conductive paint is non-toxic, water-based and water-soluble, and can therefore be cleaned easily with soap and water.

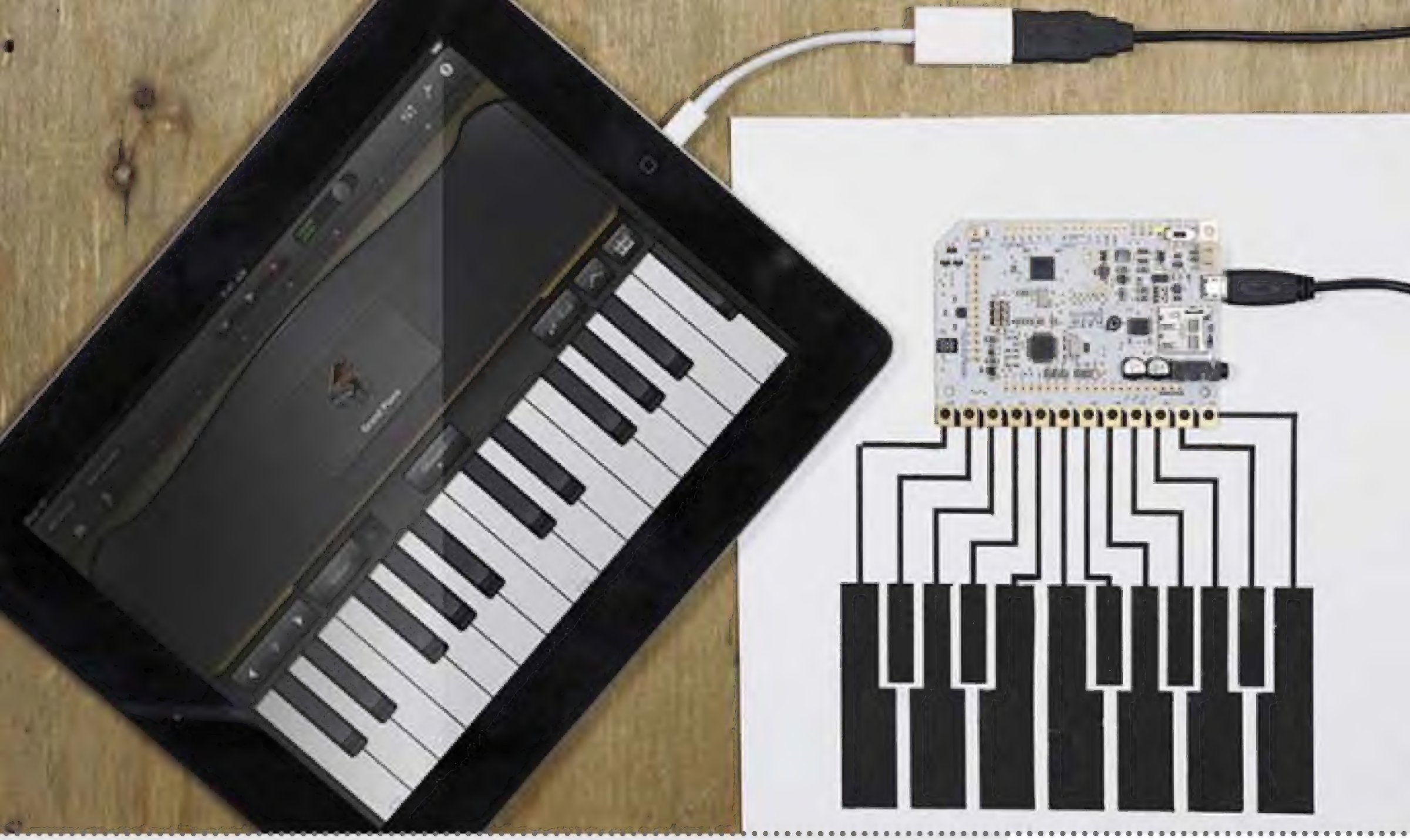
07 Make it waterproof

This paint only comes in black and is not waterproof. However, the great thing is that you can use it underneath or alongside any regular paints, varnishes and waterproofing sprays in order to act as insulation – or just to add some colour into your designs!

08 Touch and sound

Bare Conductive paint can also be used as a capacitive surface, meaning you can use it for touch, gesture

Above The ability to glue conductive materials together means you can really get creative with this



Above Yep – you can even paint a piano and hook it up to a touch control board!

or proximity controls when it is paired with a suitable control board. Bare Conductive makes its own called the Touch Board, which has everything you need to start experimenting with touch and sound. It can even act as a MIDI controller, an interface or an instrument! Bare Conductive's Touch Board is an Arduino device compatible with any existing shields and code you might have, and it works with any conductive material as well as the paint. Using the Conductive paint, you can also create touchless sensors – for example, you can draw and program an electric drum kit that responds to waves over your custom shapes.

"You can draw and program an electric drum kit that responds to waves over your custom shapes"

Secure your Raspberry Pi

Concerned about your data? Protect yourself with passwords, firewalls and some physical security

There is a distinct security risk around your Raspberry Pi. Storing anything from passwords to firewalls, this important saved data can be stolen or pocketed with minimal effort if someone knows how. So it's a relief to know that several tools and tricks can be applied to keep your device and data away from prying eyes. You might, for example, be running a home security cam with images uploaded to a cloud account. These images would be visible to anyone who possesses your Raspberry Pi's login details if you haven't bothered to change the defaults. Such a project also demands that a firewall is installed for further security on a network. Whether you're simply changing passwords, keeping your Pi under lock and key or installing a firewall, you'll be surprised at how easy it is to secure your Raspberry Pi.

THE PROJECT ESSENTIALS

Velcro

Adhesive putty

Lockable cupboard, strongbox, etc.

01 Stop using the default password

Everyone who uses a Raspberry Pi knows that the default Raspbian credentials are 'pi' and 'raspberry'. Naturally, this means that anyone can sign into your computer if you haven't changed these defaults – something you'll need to do as a matter of urgency. After signing in, open up the terminal and then set a new password with:

```
sudo passwd
```

02 Change password with raspi-config

If you're setting up a new installation of Raspbian, changing the password is one of the first things that you should do. With a new install, the first boot will automatically run the raspi-config screen.

> "Everyone who uses a Raspberry Pi knows that the default Raspbian credentials are 'pi' and 'raspberry'"

Here, use the arrow keys to find the second option, 'Change User Password', and then follow the on-screen prompts to set yourself a new passcode.

03 Create a new user account

To completely baffle anyone attempting to gain access using default credentials, take the most secure option and create a new user account. In the command line, enter:

```
sudo useradd -m username -G sudo
```

The –m switch creates a new home directory, while the second sudo adds the new account to the superuser group.

04 Give the new account a password

With the new account set up, the next step is to set a password. Since you're not signed into the account at this stage, you won't be using the passwd command. Instead, enter:

```
sudo passwd username
```

With the new account ready to use, you should now be ready to remove the default 'pi' account from Raspbian altogether.

05 Delete the default Raspbian account

You no longer need the default user account, 'pi'. Sign out and log in to your new account, and confirm it is correctly set up by opening:

```
sudo visudo
```

...and adding...

```
username ALL=(ALL) NOPASSWD: ALL
```

...to the final line. Save and exit with Ctrl+X. Now that's done, simply delete the old account with:

```
sudo deluser pi
```

Proximity sensor

If you're genuinely concerned about your Raspberry Pi's physical security, consider employing some additional hardware to make it less of a target. Your best option is probably a proximity sensor configured to detect an unauthorised presence. When coupled with a buzzer, this can detect the presence of an intruder and alert you. You can even configure an alert as an email message if you're likely to be elsewhere.

```
 GNU nano 2.2.6              File: /etc/sudoers.tmp                      Modified

Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:$

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
pi ALL=(ALL) NOPASSWD: ALL
atomickarma ALL=(ALL) NOPASSWD: ALL
```

Then remove the home directory:

```
sudo deluser -remove-home pi
```

06 Recover a lost password

If you've somehow forgotten your Raspberry Pi user account password or suspect that someone has changed it, what can you do? With a desktop computer and SD card reader, there is a way that you can recover your password. Begin by inserting the Pi's SD card into your PC's card reader.

07 Edit cmdline.txt

Find the file cmdline.txt and open it in your Linux desktop text editor. Add the following to the end of the last line of the file:

```
init=/bin/sh
```

As the Raspberry Pi boots, this command will be read, enabling us to access a screen to reset the password. Save and eject the card.

08 Change the lost password

Unfortunately, you won't be able to use SSH to recover the password, so instead connect a monitor and

Left The sudoers list gives you full control over the access privileges of users

"With a desktop computer and SD card reader, there is a way that you can recover your password"

keyboard to your Raspberry Pi. Boot the Pi and wait for the prompt, at which point you should enter:

```
passwd username
```

Type the password, hit Enter and then type it once again to confirm.

09 Initialise the Raspbian boot

Thanks to the added code, we have changed the standard Raspbian boot to display a new prompt that will let us change the password. Once you have done this, enter the following command to put everything back in order:

```
sync
exec /sbin/init
```

The Pi will now boot Raspbian normally, enabling you to sign in with the new password.

10 Revert cmdline.txt

We are not done yet, though. Safely shut down your Raspberry Pi with:

```
sudo shutdown -h now
```

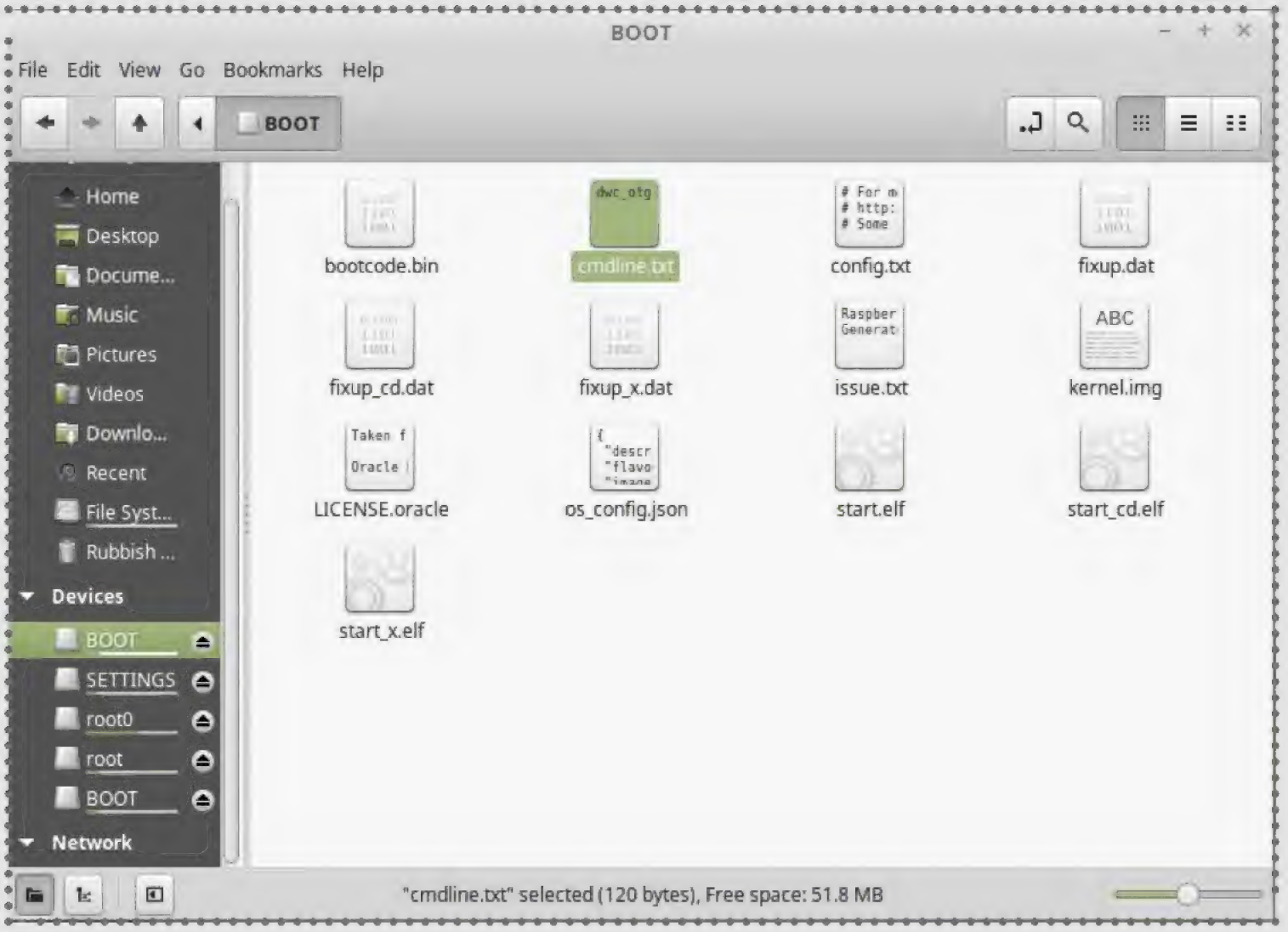
With the Pi powered down, remove the SD card and insert it into the card reader again. Open cmdline.txt in your text editor once again and remove "init=/bin/sh", then save and exit. This stops anyone else from resetting your password.

11 Physically secure your Raspberry Pi

Keeping digital intruders out of your Raspberry Pi with firewalls and secure account passwords is only part of the story. To fully protect your Pi you need to think outside of the box. Barely larger than a credit card, the Raspberry Pi computer can easily be picked up and palmed. Physical security is paramount, but a genuinely

Hiding hardware

Putting your hardware out of sight and/or reach is a good option for security, and for something as small as the Pi you have quite a few options. For instance, using Velcro or some adhesive putty you could attach the Pi to the back of a cupboard or unit, kitchen kickboards or even under a car seat. The SD card, meanwhile, is so compact that you could easily store that under a carpet or even make a home for it in a cushion or shelf – just don't lose it!



Left This script inside the /boot folder is indispensable if you lose your password

secure Raspberry Pi case – for example, one compatible with Kensington locks – has yet to be released. However, the ProtoArmour aluminium case you can get from **www.mobileappsystems.com** can be screwed to a secure surface, which is great for more permanent project setups, especially public installations.

12 Lock it in a drawer

Probably the best way to keep your Raspberry Pi secure is to make sure that you keep it locked inside a drawer or cabinet – this is particularly useful if you use the device as part of a security cam system or as a cloud server storing valuable documents, and to prevent theft in public areas like your shop or office. If no lockable storage is available and you're taking some time away from home, where it isn't practical to take the Pi with you, another solution is needed. You could travel with your Pi's SD card in your pocket, leaving the board itself creatively hidden somewhere with Velcro.

"Keep it locked inside a drawer or cabinet – this is particularly useful if you use the device as part of a security cam system"

Left If you tend to SSH into your Pi anyway, hiding it away is a great option

13 Add a firewall

Regardless of which operating system you're using, adding a firewall is a guaranteed way to improve your computer's security. While the Raspberry Pi has a built-in firewall, it is tricky to configure. Thankfully, some other people have noticed this too and released fwbuilder, an interface to the otherwise complex iptables firewall that comes with Raspbian.

14 Install fwbuilder in Raspbian

Because iptables is a bit fiddly and errors can leave you with no network connection, fwbuilder has been developed to make firewall configuration quick and painless. We'll use the apt-get command to first check for updates and then install fwbuilder:

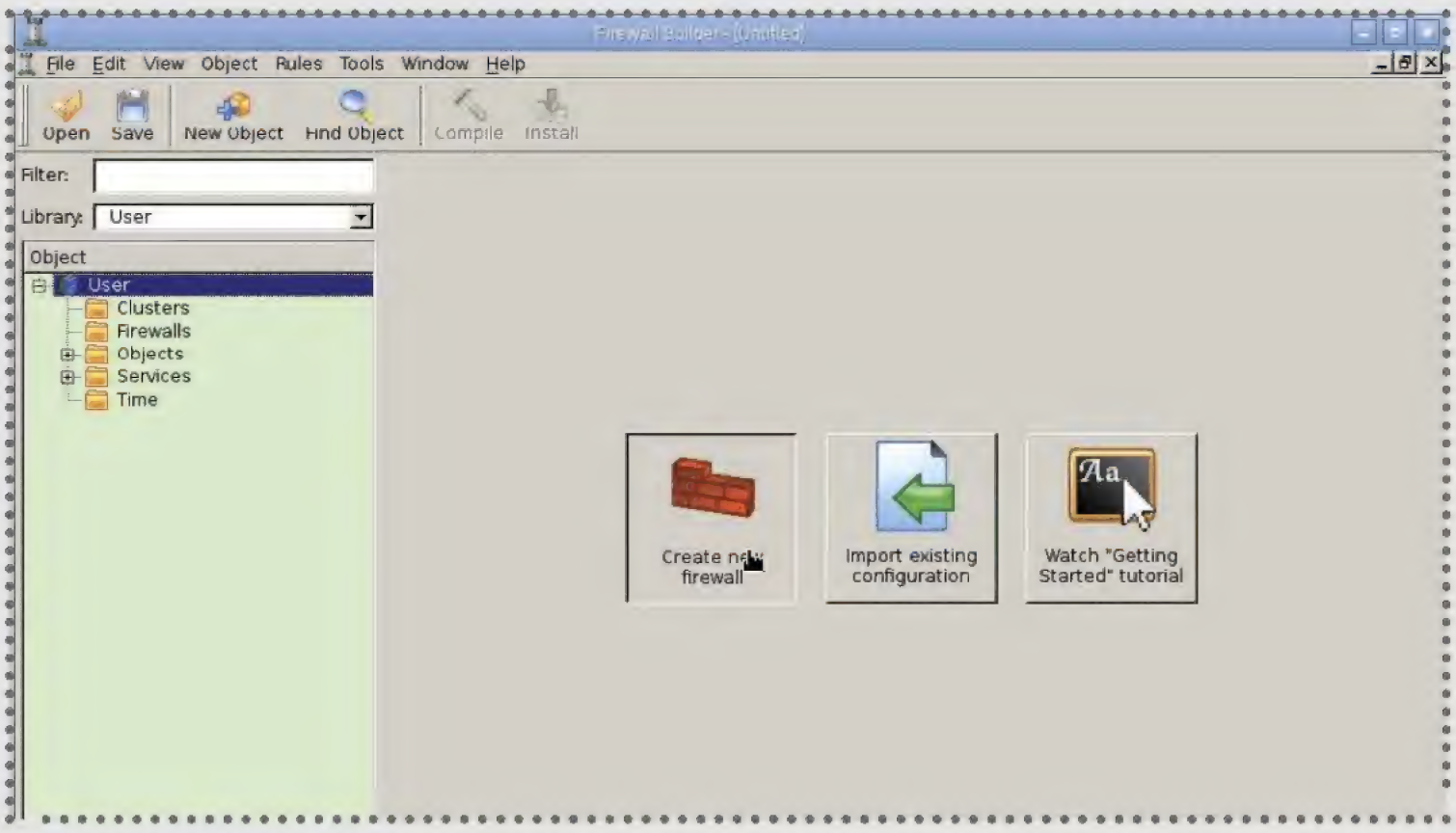
```
sudo apt-get update
sudo apt-get install fwbuilder
```

Follow the prompts to install and, once complete, switch to the Raspberry Pi GUI by entering:

```
startx
```

"Fwbuilder is an interface to the otherwise complex iptables firewall that comes with Raspbian"

In the Pi's graphical desktop, launch fwbuilder from the Internet menu. Upon launching fwbuilder, follow the given steps to set up your Raspberry Pi firewall and save the resulting script. We're nearly done now, but some adjustments are still required before your Pi fully connects to the network.



Left Fwbuilder makes iptables more accessible by providing a GUI

15 Complete firewall configuration

Launch the /etc/network/interfaces script in your text editor and complete configuration by adding:

```
pre-up /home/pi/fwbuilder/firewall.fw
```

Next, find the section labelled "Epilog" and add:

```
route add default gw [YOUR.ROUTER.IP.HERE] eth0
```

If you're using a wireless card, add the same line but switch the last characters to wlan0:

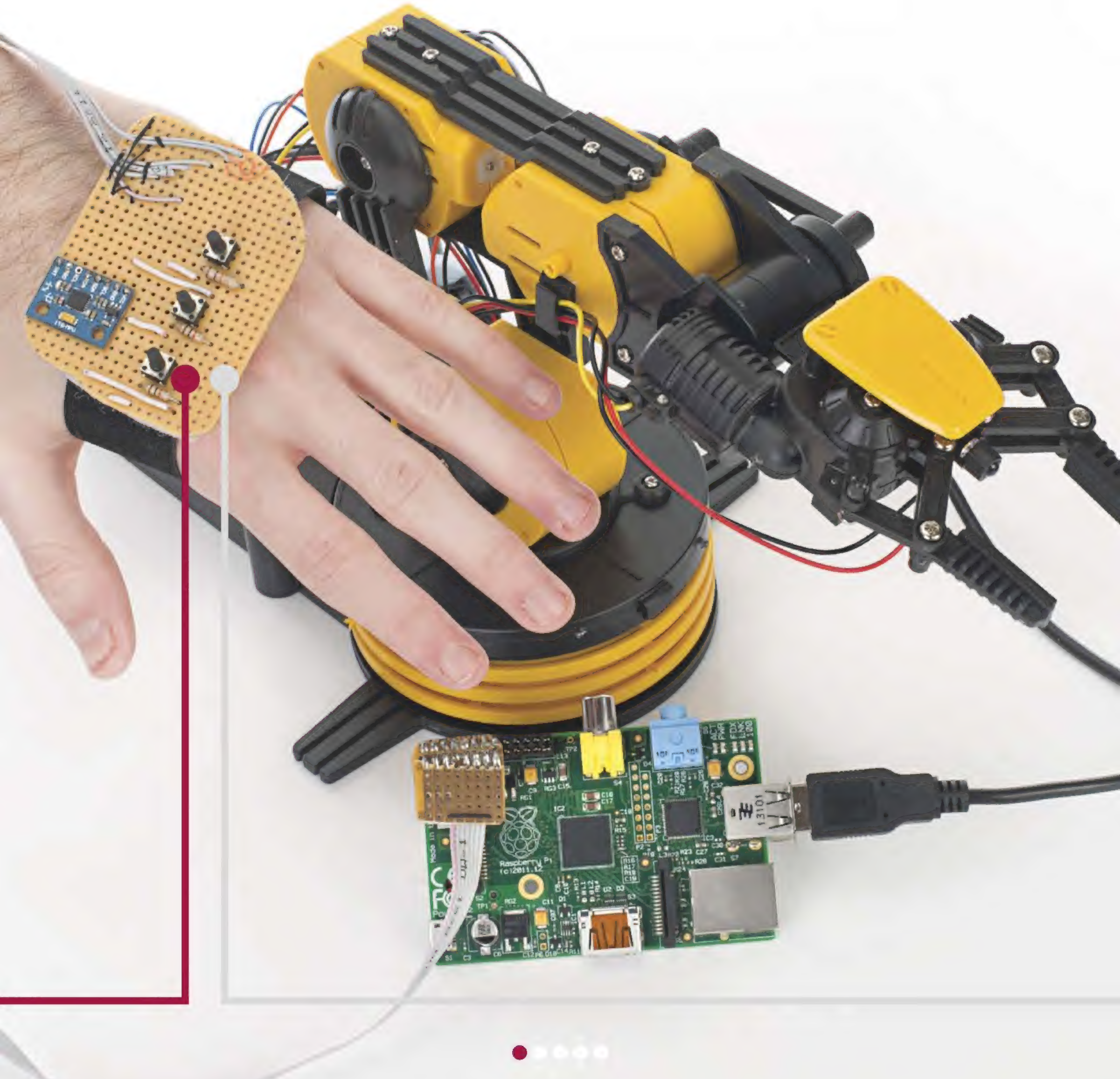
```
route add default gw [YOUR.ROUTER.IP.HERE] wlan0
```

16 Back up your data

While losing your Raspberry Pi or the data on it might initially seem like a disaster, don't be disheartened. As long as you have taken steps to back up your data or clone your SD card, you will at least have continuity when you resume the project. Backing up is crucial!

Robot Arm

Joseph Thomas gets to grips with natural motion control for his Raspberry Pi-powered robot arm



What first inspired you to begin your robot arm project?

The robot arm itself was one I'd seen years ago, and I really wanted it because it's something that you can control yourself – it really captured my young imagination. I was volunteering at a science museum down here in Harlow and this club based around the Raspberry Pi sprung up, and I bought the robot arm because I wanted it. So then I had the Raspberry Pi thing going on at the same time and thought, why not meld the two?

I had this complicated system of key presses to get it to do anything, which was a bit boring, and then James Dali (one of the people who helps out with the club) gave me the idea of shoving an accelerometer on the top of it to give an idea of where it is. I took that and thought, 'What if I had the accelerometer on me and sort of used it to mirror the motion of my hand?' So I looked around, searched up the accelerometer he was using (the MPU-6050) and then found it for about £5 on eBay – it's normally about £30 from SparkFun but I'm on a student budget... A lot of the code I've used is borrowed but open source, and people have said it's fine, so then I went through and had two programs – one that could control the arm, one that took the input in from the accelerometer – and kind of just smushed them together. It's not that nice to look at, but it works and that's all that really matters.

So what exactly are you reading with that MPU-6050?

There's the gyroscope and the accelerometer in the code I'd found – you can use one or the other, but the gyroscope is very good for degrees over time and it tends to drift, while the accelerometer is good for sudden turns and for measuring gravity. If you compare the two to each other then you can get a rough angle all of the time, so it's essentially

Joseph Thomas is a student helping to run a Raspberry Pi club from a science museum in Harlow, where they have worked on projects ranging from a robot arm to a portable Pi.

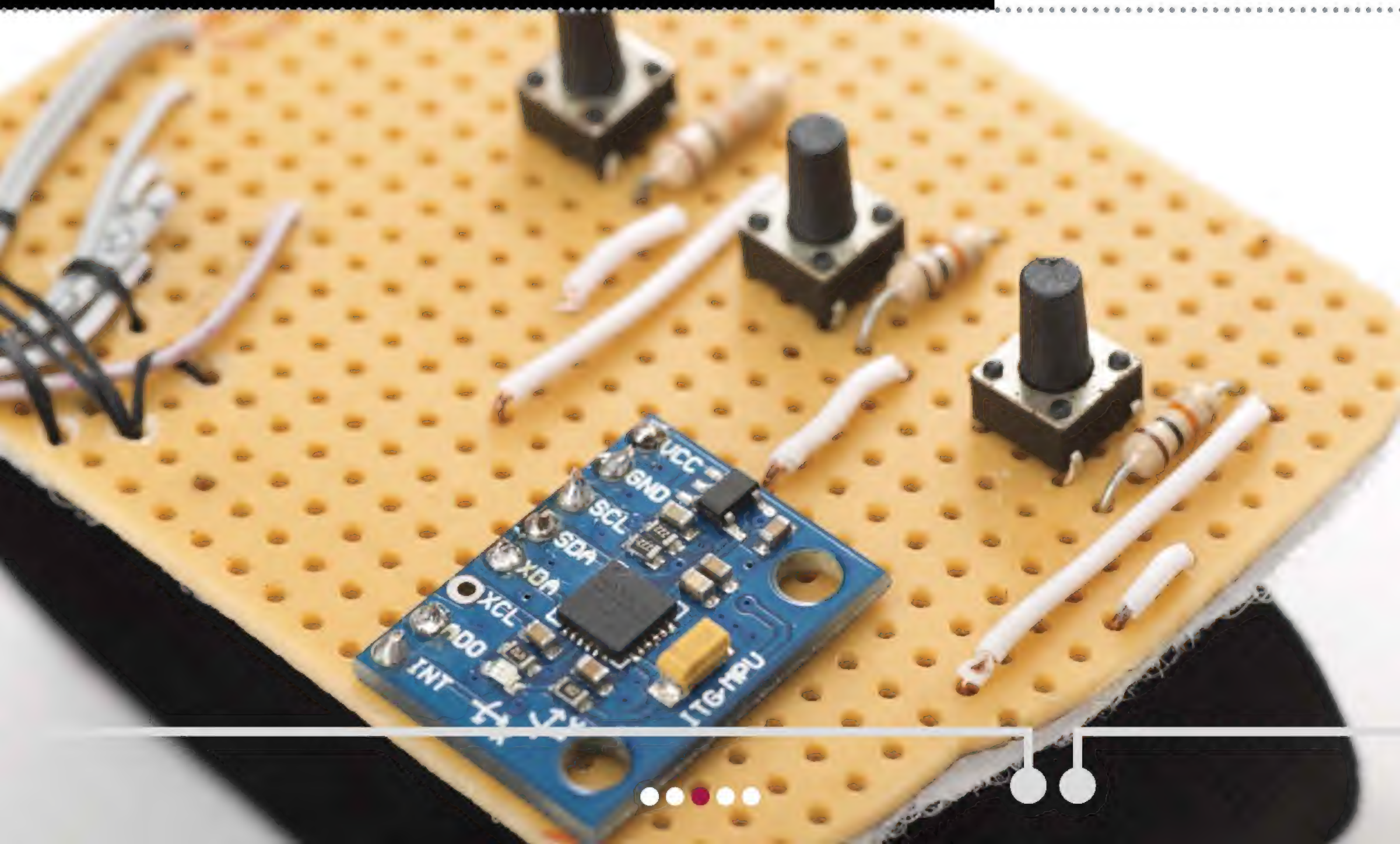
the accelerometer and the gyroscope used together to correct the faults with one or the other. It's got two axes of motion – pitch and roll.

Take us through the code itself.

So in the first bit it finds where the actual I2C interface is and there's a quick setup – I've got three buttons on there to control the gripper and the lights, so it sets those up – and then there's a bit which is using the USB library to find the robot arm, then spitting it out if that's an issue. There are a couple of definitions for some functions to actually move the arm, so it's a little bit easier – each motor direction is a different binary number – and then there are more definitions for setting up reading data from the accelerometer and a bit of maths for making sure the gyro and the accelerometer are both giving the correct angle. Then there's this while loop with a try inside it that is just pulling the accelerometer for data, spitting out the maths

If you like

The robot arm that Joseph is using can be bought from Maplins in the UK (**bit.ly/1Da9BrT**) or ordered from Adafruit any where else in the world (**bit.ly/1yXlDQt**). There are many guides online to get you up and running, like this: **bit.ly/1AKd0OU**.



stuff, before just checking that the angle given is within a certain range. If it is, move this motor left (for example), or if a button is pressed then it turns a light on. The only problem I've had with it is that to actually move it, it requires a change in angle – so there's not a continuous thing. I have to wave my hand a little bit, but there's that degree angle and if I trip it then it'll move around.

Have you considered adding any more forms of control for the arm?

Yeah, I've done a lot of research into this. In terms of other ways to control it, I quite like the intuitiveness of it – to rotate and move this arm you are moving your own arm, so that's something I've been focussing on and trying to get even more intuitive. Trying to get some sort of – I bought an Arduino at some point – trying to build an actual robotic hand and then spreading out from there. Eventually, my big plan – many, many years in the future – is to have an entire sort of human body that is controlled by the movements of the user, but that's a very large plan which I haven't put too much into just yet! But essentially, the prototype that people have done before is sort of having pot sensors – potentiometers – on the fingers just to measure the actual rotation and closing of the fist, then having that represented with servos and then possibly doing that with actual pieces of string to sort of emulate the tendons. So you'd have a single servo, or a couple of servos, in an arm bit that would pull string which would close each finger in turn.

Another idea, which seems to be one of the most viable, is having it completely brain-controlled... There's a fair amount of interest in reading brain activity – you can do it with the NeuroSky, for example. There's quite a nice open source project which I might end up using because it has

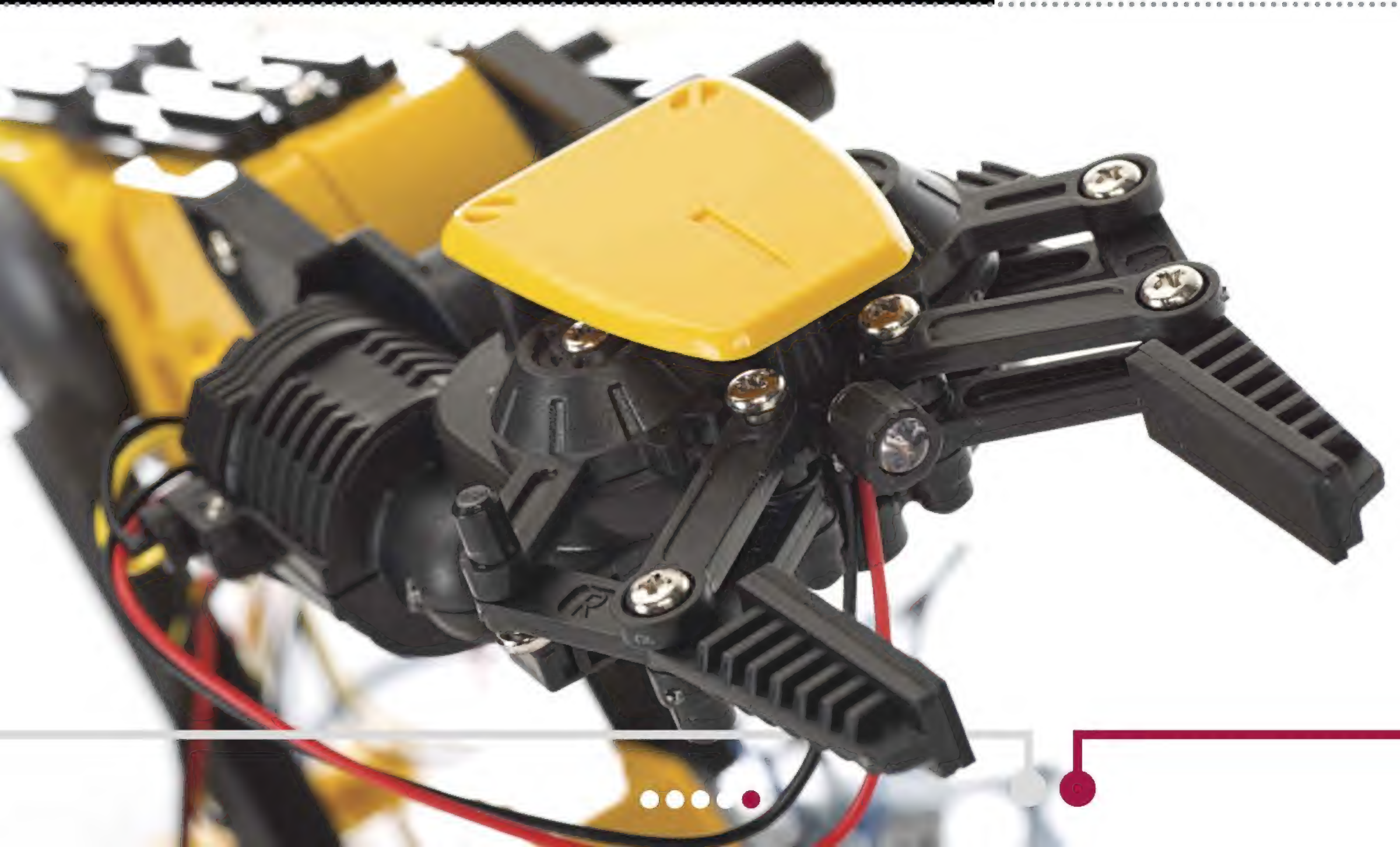
Further reading

NeuroSky has a whole product family dedicated to EEG and ECG biosensors, including the popular MindWave headsets (**neurosky.com**), and there are a few hacks available too (**bit.ly/1C7w0SP**). OpenBCI is a burgeoning open source project dedicated to brain-computer interfaces (**openbci.com**).

four inputs, so you can measure at least two things at once, and that seems to be a fairly interesting place to go. It's expensive though, and if you're going open source then they have a lot of warnings on the websites saying that you do this at your own risk, this is not a medical product, you may fry your brain...

What is the next step then?

Further projects would probably be replacing the motors. Because it's motor-driven, it's timing-based, so having something with servos instead where I can have a definite angle would be a lot more useful, a lot more precise and wouldn't tend to go... one of the problems with it is that if you tell it to keep going in one direction, it will keep going in one direction whether it wants to or not, and there's this awful grinding of gears as it attempts to go in one direction and can't. So that will probably be a new arm, a new robot, trying to get it a bit more nice-looking and more precise.



Supercharge your Pi

Get the most out of your Raspberry Pi with these performance-enhancing tips and tricks

Your Raspberry Pi is plugged in. Raspbian is installed on the SD card and you are right in the middle of setting up a wireless print server. But are you truly getting the most from your little computer? Do the components you're using maximise the potential of your Raspberry Pi or are they holding it back? Perhaps you haven't explored the full set of options in Raspbian, or you're running the entire OS from an SD card, something that can reduce SD card lifespan. Various techniques can be employed to improve performance, from choosing the right hardware to overclocking the CPU. You might even maximise storage space on the Raspberry Pi's SD card or all but replace it with a secondary device to improve speed. Here are some tips and tricks for optimising your Pi's performance.

"A low capacity SD card with poor error correction is going to be slower than a larger card with greater resilience"

01 Use better storage hardware

Your choice of storage media can have an impact on your Raspberry Pi's performance, regardless of the operating system. A low capacity SD card with poor error correction is going to be slower than a larger card with greater resilience, so you need to find the right balance for your project and shop wisely.

02 Choosing the best SD card

Various standards of SD card are available, with the more expensive designed for better error correction. For the best performance on your Raspberry Pi, choose an SDHC card with a high rating. The same advice applies to MicroSD cards, which you can use on your old Raspberry Pi with an SD card adaptor or directly insert into a Raspberry Pi B+ or 2B.

Above There are some good third-party options, but the official SDs that come with NOOBS have brilliant performance

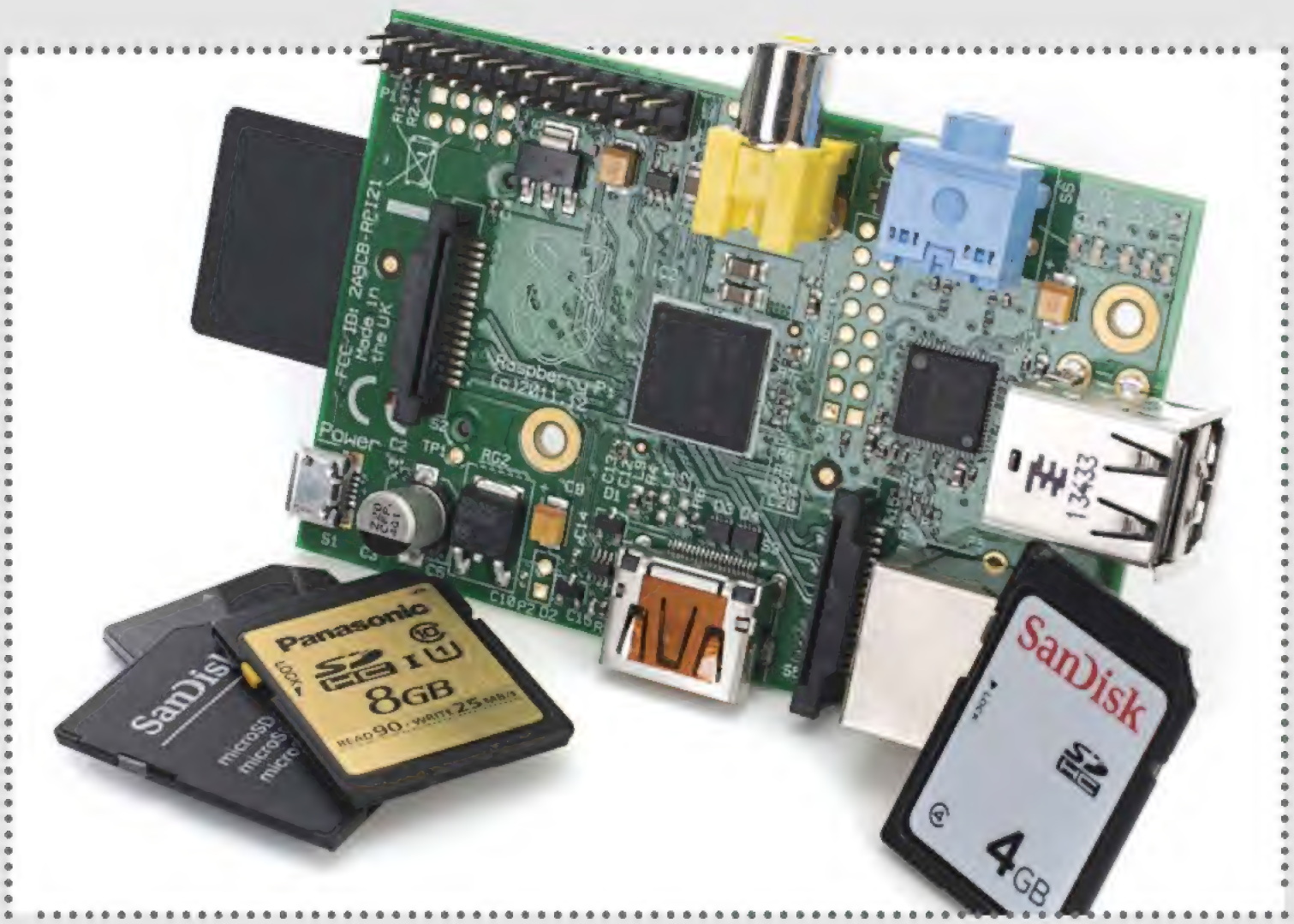
03 Make the most of your storage

You'll typically need 1-2GB of storage for your chosen Raspberry Pi distro – and to run Ubuntu MATE you'll need at least 4GB – so any remaining storage on your SD card will be used for updates and data you create or save. In Raspbian, you can open a command line and run the configuration utility to gain more space, but only if your SD card's greater than 2GB:

```
sudo raspi-config
```

04 Expand the Raspbian partition

Maximising the partition affords the full capacity of your SD card, which will increase the media's lifespan – there is more space to write too, so the same sectors aren't being overwritten as often. With raspi-config running, use the arrow keys to select expand_rootfs in the menu. After that, just wait briefly while the partition is resized.



Left For more info on the best SD cards to use, head over to **http://bit.ly/1M48TUa**

05 Write data to RAM

Rather than reading and writing data to your SD card – something that will eventually result in a deterioration of reliability and performance – you can configure Raspbian to write to the system RAM, which will speed things up slightly and improve SD card performance. This is achieved using fstab (file systems table), a system configuration tool available in most Linux distros.

06 Enable fstab in Raspbian

This is much like creating a RAM disk in Windows and is almost as easy to setup. In the command line, enter:

```
sudo nano /etc/fstab
```

Add the following line to mount a virtual file system:

```
tmpfs /var/log tmpfs defaults,noatime,nosuid,m
ode=0755,size=100m 0 0
```

Follow this by saving and exiting nano (Ctrl+X), then safely restarting the Pi:

```
sudo shutdown -r now
```

"You can configure Raspbian to write to the system RAM, which will speed things up slightly and improve SD card performance"

07 Configure fstab for fast performance

Upon restarting, the virtual file system will be mounted and /var/log will be on the RAM disk. Other directories that can be moved to RAM include:

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,size=100m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid,size=30m 0 0
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=100m 0 0
tmpfs /var/run tmpfs defaults,noatime,nosuid,mode=0755,size=2m 0 0
tmpfs /var/spool/mqueue tmpfs defaults,noatime,nosuid,mode=0700,gid=12,size=30m 0 0
```

Add each to /etc/fstab using nano.

08 Move your OS to a HDD

If you're concerned about the lifespan of the SD card, why not reduce your Raspberry Pi's reliance on it? Instead of using the SD card as a sort of budget SSD, change its role and add a HDD or USB stick to run the operating system, leaving the SD card for bootstrapping. This can give a marked performance boost to the SD card.

09 Back up the SD card

Begin by creating a copy of your Raspberry Pi's SD card. Shut down, remove the card and insert it into your desktop computer. In the command line, run:

```
sudo dd bs=4M if=/dev/sdb of=~/backup.img
```

The path /dev/sdb represents the SD card. Copying should take 5-10 minutes. Once complete, remove the SD card and connect your USB device.

Picking a USB drive

Speeding up your Raspberry Pi by migrating the root filesystem to an external USB drive is a start, but what sort of device should you use for the best performance? With a USB thumb drive you can add flash storage up to 16GB without running into any significant problems (the larger the drive, the greater the current required to read/write). Anything larger is expensive and unnecessary.

10 Copy Raspbian to USB

Using a blank ext4-formatted USB thumb drive (or external HDD) as the destination drive, enter:

```
sudo dd bs=4M if=~/backup.img of=/dev/sdc
```

Leave the backup on your computer, just in case something goes wrong. With an SD card and USB storage device sharing an identical disk image, it's time to consider what you're going to do next – create a faster Raspberry Pi.

11 Split the Raspbian partitions

Ideally, the boot partition should remain on the SD card while the root filesystem is run from the external HDD or USB thumb drive. Using your preferred partition manager (Disk Utility is in most distros), unmount and delete the root filesystem from the SD card, ensuring you have retained the boot partition. After removing the SD card, connect your USB device and delete the boot partition, taking care to leave the root filesystem intact. Then resize the root filesystem on the USB device, making sure that 10MB remains.

"Ideally, the boot partition should remain on the SD card while the root filesystem is run from the external HDD or USB thumb drive"

12 Identify the root filesystem

With this configuration, you're going to have the SD card and the external USB storage connected, so you need to tell the Pi where the root filesystem is. Still on the desktop Linux computer with your SD card inserted, run:

```
sudo nano /boot/cmdline.txt
```

Find "root=/dev/mmcblk0p2" (or similar) and change that to read "root=/dev/sda2", which is your external USB storage. Save and exit.

13 Add other USB devices

You can now restart your Raspberry Pi with the storage devices attached, but as soon as you connect further USB media you'll suffer problems. Avoid this by installing gdisk:

```
sudo apt-get update
sudo apt-get install gdisk
```

Then run gdisk:

```
sudo gdisk /dev/sdb
```

Enter '?' to display the options and select 'Recovery and Transformation options (experts only)', followed by 'Load MBR and Build Fresh GPT'. Tap '?' one last time and select 'Write Table to Disk', then exit. Remove and replace the USB device and run gdisk again. This time, enter 'i' then '1' to display the Partition Unique GUID.

14 Make your Pi fast & reliable

Make a note of the GUID and then switch to the SD card. Reopen cmdline.txt and change "root=/dev/mmcblk0p2" to "root=PARTUUID=XXXXXX", where the numerical string from the partition unique GUID should replace the "XXXXXX". When you're done, save and exit. You

"As soon as you connect further USB media you'll suffer problems. Avoid this by installing gdisk"

can then start your Raspberry Pi. Congratulations, your Raspberry Pi is now faster and more reliable to use!

15 Boost performance with overclocking

Need more from your Raspberry Pi? It is possible to overclock the computer, although you should be aware of the risks inherent with this activity. You should also ensure that your Raspberry Pi's processor is suitably cooled – heatsinks for the CPU, Ethernet controller and power regulator can be purchased online.

16 Overclock your Raspberry Pi

Overclocking is available through raspi-config. Launch from the command line and arrow down to the overclock option. Four further options are available: Modest, Medium, High and Turbo. With your ideal clock speed selected, exit raspi-config and restart your Raspberry Pi to apply the changes:

```
sudo shutdown -r now
```

Now you will need to perform tests to see how stable it is overclocked. Raspberry Pi founder, Eben Upton, suggests running *Quake 3* as a good stress test. Should the Pi fail to boot, hold Shift to boot without overclocking, run raspi-config and select a more modest overclock.

17 Run Raspbian without the GUI

Despite these changes, you may find that the GUI remains slow. If you run a lot of commands in bash, the best thing to do is disable launching into X. In raspi-config, choose "boot_behaviour" and select the first (default) option to ensure your Pi boots to the command line. If you need the GUI, enter `startx` in the terminal.

Overclock with a heatsink

Overclocking is potentially dangerous to any computer system, which is why it's great that the Raspberry Pi developers have included the facility in their approved operating system and allowed its use under warranty. If you're using this feature, heatsinks and water cooling systems are available for the Raspberry Pi to ensure you don't bake the CPU and RAM when in use.

Create your own digital assistant – part one

Everyone would like to tell their computer exactly what to do. Well, with Python and a Pi, now you can

"While J.A.R.V.I.S. has massive amounts of computing power behind him, you can construct the front-end with very modest resources"

Everyone who has watched the *Iron Man* movies has probably dreamed of having their own artificially intelligent computer system to do their every bid and call. While J.A.R.V.I.S. has massive amounts of computing power behind him, you can construct the front-end with very modest resources. With a Raspberry Pi and the Python programming language, you can build your own personal digital assistant that can be used as a front-end to whatever massive supercomputing resources that you use in your day-to-day life as a playboy, philanthropist genius. We will go over the basics that you will need to know over the next few issues, so that by the end of the series

you should be able to build your own rudimentary, customised agent.

The first step to interacting with the humans around us is to listen for verbal commands so that we know what we need to process. You have several options available to handle this task. To keep things simple, we will be dealing only with devices that are plugged into one of the USB ports. With that stipulation, you can then talk directly with the USB device at the lowest level. This might be necessary if you are trying to use something that is rather unusual to do the listening, but you will probably be better off using something that is a bit more common. In this case, you can use the Python module PyAudio. PyAudio provides a Python wrapper around the low level cross-platform library PortAudio. Assuming that you are using something like Raspbian for your distribution, you can easily install the required software with the command:

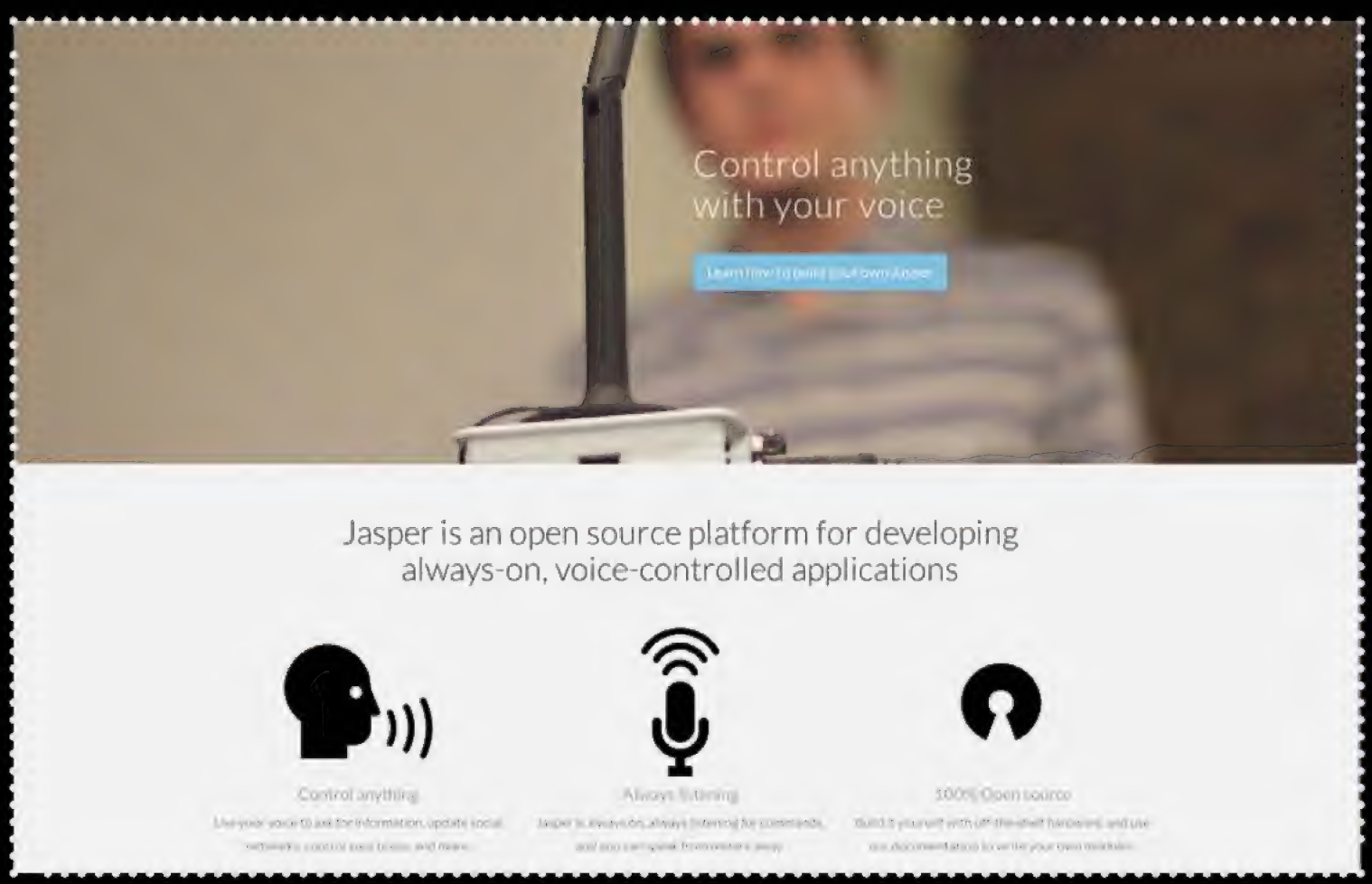
"To keep things simple, we will be dealing only with devices that are plugged into one of the USB ports. With that stipulation, you can then talk directly with the USB device at the lowest level"

```
sudo apt-get install python-pyaudio
```

If you need the latest version, you can always grab and build it from source. PyAudio provides functionality read in audio data from a microphone, along with the ability to play audio data out to your headphones or speakers. So we will use this as our main form of interaction with the computer.

The first step is to be able to read in some audio commands from the humans who happen to be nearby. You will need to import the 'pyaudio' module before you can start interacting with the microphone. The way PyAudio works is similar to working with files, so it should seem familiar to most programmers. You start by creating a new PyAudio object with the statement p

= pyaudio.PyAudio(). You can then open an input stream with the function p.open(...), with several parameters. You can set the data format for the recording; in the example code we used format=pyaudio.paInt16. You can set the rate in Hertz for sampling. For example, we are using rate=44100, which is the standard 44.1KHz sampling rate. You also need to say how big a buffer to use for the recording – we used frames_per_buffer=1024. Since we want to record, you will need to use input=true. The last parameter is to select the number of channels to record on; in this case we will use channels=2. Now that the stream has been opened, you can start to read from it. You will need to read the audio data in using the same chunk size that you used when you created the stream – it will look like stream.read(1024). You can then simply loop and read until you are done. There are then two commands to shut down the input stream. You need to call stream.stop_stream() and then stream.close(). If you are completely done, you can now call p.terminate() to shutdown the connection to the audio devices on your Raspberry Pi.



Left If you want a full virtual assistant rather than just voice control, check out Jasper: **http://bit.ly/1g7DWMU**

The next step is to be able to send audio output so that Jarvis can talk to you as well. For this you can use PyAudio, so we won't have to look at another Python module. To make things simple, let's say that you have a WAVE file that you want to play. You can use the 'wave' Python module to load it. Once again, you will create a PyAudio object and open a stream. The parameter 'output' should be set to true. The format, the number of channels and the rate is all information that will be derived from the audio data stored in your WAVE file. To actually hear the audio you can simply loop through, reading one chunk of data from the WAVE file at a time and immediately writing out to the PyAudio stream. Once you're done, you can stop the stream and close it.

In both of the above cases, the functions block when you call them until they have completed. What are the options if you want to still be able to do processing while you are either recording audio or outputting audio? There are non-blocking versions that take a callback function as an extra parameter called stream_callback. This callback function takes four parameters, named in_data, frame_count, time_info, and status. The in_data parameter will contain the recorded audio if input is true. The callback function needs to return a tuple with the values out_data and flag. Out_data contains the data to be outputted if output is true in the call to the function open. If the input is true instead, then out_data should be equal to None. The flag can be any of paContinue, paComplete or paAbort, with obvious meanings. One thing to be aware of is that you cannot call, read or write functions when you wish to use a callback function. Once the stream is opened, you simply call the function stream.start_stream(). This starts a separate thread to handle this stream processing. You can use stream.is_

"To actually hear the audio you can simply loop through, reading one chunk of data from the WAVE file at a time and immediately writing out to the PyAudio stream"

active() to check on the current status. Once the stream processing is done, you can call stream.stop_stream() to stop the secondary thread.

Now we've covered how to get audio information into and out of your Raspberry Pi, you can start by adding this functionality to your next project. In the next issue, we will look at how to convert this audio information into something usable by the computer by using voice recognition modules. We will also look at the different ways to turn text into audio output using TTS modules.

The Code DIGITAL ASSISTANT

```
# You need to import the pyaudio module
import pyaudio

# First, we will listen
# We need to set some parameters
# Buffer chunk size in bytes
CHUNK = 1024
# The audio format
FORMAT = pyaudio.paInt16
# The number of channels to record on
CHANNELS = 2
# The sample rate, 44.1KHz
RATE = 44100
# The number of seconds to record for
RECORD_SECS = 5

# Next, we create a PyAudio object
p = pyaudio.PyAudio()

# We need a stream to record from
stream = p.open(format=FORMAT, channels=CHANNELS,
                rate=RATE, input=TRUE, frames_per_buffer=CHUNK)
```

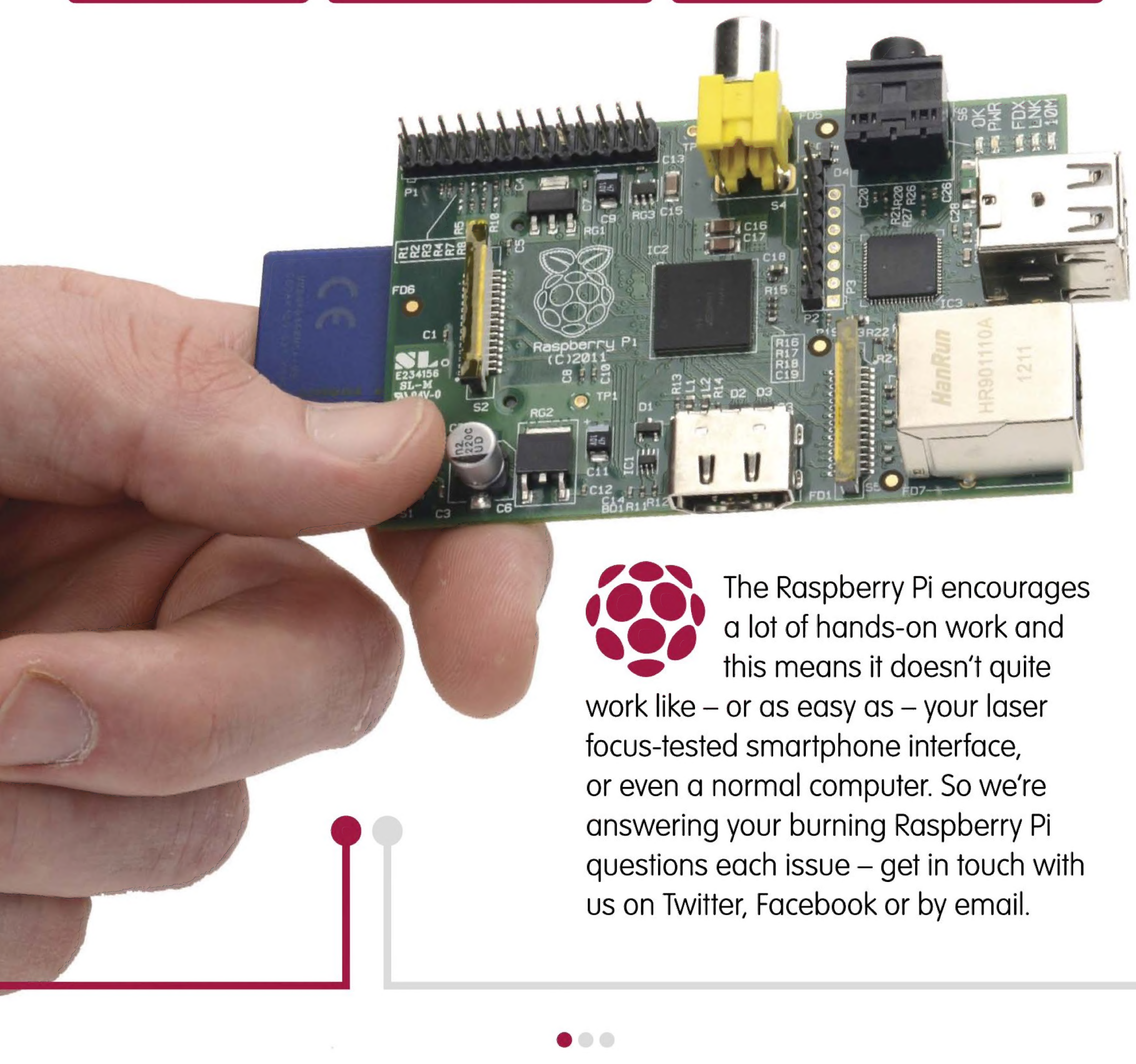
The Code DIGITAL ASSISTANT

```
# We can now record into a temporary buffer
frames = []
for i in range(0, int(RATE / CHUNK * RECORD_SECS)):
    data = stream.read(CHUNK)
    frames.append(data)

# We can now shut everything down
stream.stop_stream()
stream.close()
p.terminate()

# If we want to play a wave file, we will need the wave module
import wave

# We can open it, give a filename
wf = wave.open("filename.wav", "rb")

# We need a new PyAudio object
p = pyaudio.PyAudio()

# We will open a stream, using the settings from the wave file
stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                channels=wf.getnchannels(), rate=wf.getframerate(), output=True)

# We can now read from the file and play it out
data = wf.readframes(CHUNK)
while data != '':
      stream.write(data)
      data = wf.readframes(CHUNK)

# Don't forget to shut everything down again
stream.stop_stream()
stream.close()
p.terminate()
```

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@imagine-publishing.co.uk



The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

What's the best way to follow the Astro Pi project?
Aziz via email

You're in luck! Astro Pi recently got itself a shiny new website – **https://astro-pi.org** was relaunched by the Raspberry Pi Foundation a little while ago, and it now features all of the mission updates that were posted to the official blog along with all sorts of other interesting goodies. You can learn all about the mission itself, the people involved and the ISS, and there are some awesome learning resources. For a start, you can download all of the code for the winning projects, plus there are some great guides to using the Sense HAT that are brilliant for code clubs or a bit of weekend fun.

I want to make games on my Raspberry Pi – where do you think I should start?
Natalie via Facebook

Pygame Zero is the way to go here. A lot of people use Pygame to create games, and to make it even easier for people to jump in, a chap called Daniel Pope simplified Pygame even further into a library called Pygame Zero. This gives you friendlier, more readable lines of code and gets rid of the 'boilerplate' material that you'd have to include in a regular Pygame project, with the upshot being that it's much easier to learn. You can check it out for yourself over at **https://pygame-zero.readthedocs.org**.

Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

JUST A SCORE
WHAT'S YOUR JUST A SCORE?

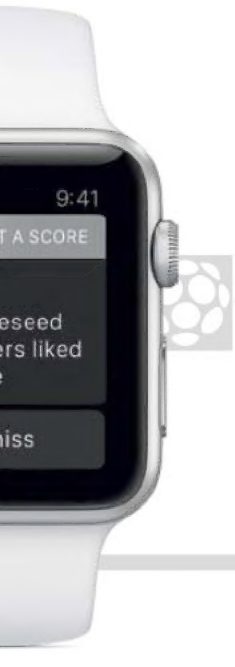
Have you heard of Just A Score? It's a new, completely free app that gives you all the latest review scores. You can score anything in the world, like and share scores, follow scorers for your favourite topics and much more. And it's really good fun!

Have you got any tips for condensing GPIO-heavy code?
Jackie via email

Try GPIO Zero. Created by Ben Nuttall, Dave Jones and a few other people, it's designed to make it simpler to work with GPIO ports and also has the effect of cutting down on the code. In the examples on the project site (**http://pythonhosted.org/gpiozero**), you can see that it replaces big while loops and repeated sleep commands with much simpler statements like button.wait_for_press(). Translate your code into GPIO Zero and see how many lines you save!

I heard that you can get a customised Raspberry Pi now?
Malcolm via email

Absolutely – although you're going to have to wait a while to get your hands on one, unless you have a fair bit of cash to spare! Element14, the official manufacturer and distributor of the Pi, has launched a customisation service in partnership with Raspberry Pi Trading (the business end of the Foundation). If you're willing to place an order for a few thousand boards, you can work with element14's design and engineering teams to make adjustments, like adding and removing components or changing the layout. The service has only just launched so we won't see any custom Pi boards until next year.



Next issue

Get inspired · Expert advice · Easy-to-follow guides

20 AWESOME PROJECTS

Made by you

Get this issue's source code at:

www.linuxuser.co.uk/raspicode